

(19)日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11)特許出願公開番号

特開平10-79042

(43)公開日 平成10年(1998)3月24日

(51)Int.Cl.<sup>6</sup>

識別記号

庁内整理番号

F I

技術表示箇所

G 0 6 T 13/00

G 0 6 F 15/62

3 4 0 A

H 0 4 N 5/262

H 0 4 N 5/262

審査請求 有 請求項の数17 O L (全 38 頁)

(21)出願番号 特願平8-233492

(22)出願日 平成8年(1996)9月3日

(71)出願人 396001980

株式会社モノリス

東京都港区麻布十番1丁目7番3号

(72)発明者 伊藤 博文

東京都港区麻布十番1丁目7番3号 株式会社モノリス内

(72)発明者 品川 嘉久

東京都江戸川区西葛西5-10-26-204

(72)発明者 國井 利▲やす▼

東京都文京区本郷1-25-21 ドムス本郷602

(74)代理人 弁理士 吉田 研二 (外2名)

(54)【発明の名称】 アニメーション処理方法およびその応用

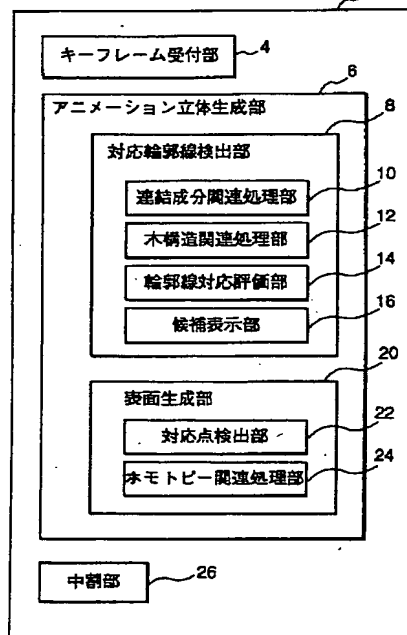
(57)【要約】

【課題】 従来のアニメーションの中割システムは位相が変化するような場合に対応できず、用途が限られていた。また、操作性も悪かった。

【解決手段】 アニメーション立体生成部6をもつ。アニメーション立体とは、高さz方向に時間tをとり、平面 $t=t_0$ による断面がその時刻におけるフレーム画像になるような立体である。この立体の生成に当たり、連結成分や輪郭線の木構造など、位相を考慮する。この立体の断面を切り出せば中割が実現する。位相幾何学的な根拠はモース理論に求める。

実施形態

2 中割システム



## 【特許請求の範囲】

【請求項1】 フレームをキーフレームとそれ以外の中間フレームに分類して処理を行うアニメーション処理方法において、

オブジェクトが描かれた複数のキーフレームを準備するキーフレーム準備工程と、

アニメーション立体を生成する工程であって、その立体の断面の輪郭線形状を前記キーフレームに描かれたオブジェクトの輪郭線形状に一致させたうえで、その立体の位相に注目しながらその立体の表面を生成する立体生成工程と、

を含むことを特徴とするアニメーション処理方法。

【請求項2】 請求項1に記載の方法において、該方法はさらに、

生成されたアニメーション立体の前記断面とは異なる断面における輪郭線形状をもとに中間フレームに含まれるオブジェクトの二次元形状を取得する補間工程を含むことを特徴とするアニメーション処理方法。

【請求項3】 請求項1に記載の方法において、

前記立体生成工程は、

隣接するキーフレームに描かれているオブジェクトの輪郭線どうしの近さをもとに隣接するキーフレーム間における輪郭線の対応関係を把握する対応輪郭線検出工程と、

対応する輪郭線間の表面をホモトピーの概念を用いて生成する表面生成工程と、

を含むアニメーション処理方法。

【請求項4】 請求項3に記載の方法において、

前記対応輪郭線検出工程は、隣接するキーフレーム間で輪郭線の数不一致なとき、それらのキーフレーム間で輪郭線の自然消滅または自然発生があったものとみなし、その自然消滅または自然発生を考慮して輪郭線の対応関係を把握するアニメーション処理方法。

【請求項5】 請求項3に記載の方法において、

前記対応輪郭線検出工程は、隣接するキーフレーム間で輪郭線の自然消滅または自然発生があった場合、輪郭線の本構造をもとに自然消滅または自然発生すべき輪郭線を推定して輪郭線の対応関係を把握するアニメーション処理方法。

【請求項6】 請求項3に記載の方法において、

前記対応輪郭線検出工程は、アニメーション立体の連結成分の数に関する指定にもとづいて輪郭線の分岐を考慮し、輪郭線の対応関係を把握するアニメーション処理方法。

【請求項7】 請求項3に記載の方法において、

前記表面生成工程は、対応する輪郭線上において対応しあう点をそれらの点の近さをもとに検出し、これらの対応しあう点をホモトピーの軌跡で連続的につなぐことで立体の表面を生成するアニメーション処理方法。

【請求項8】 請求項7に記載の方法において、

前記表面生成工程は、対応する2つの輪郭線A、Bそれぞれの上の点a、bについて、点aから見て輪郭線B上の最も近い点が点bであり、かつ点bから見て輪郭線A上の最も近い点が点aであるとき、これら点a、bを対応しあう点として検出するアニメーション処理方法。

【請求項9】 請求項3に記載の方法において、

前記対応輪郭線検出工程は、隣接するキーフレームに描かれているオブジェクトの移動による影響を差し引いたうえで、それらキーフレーム上に存在する輪郭線どうしの近さを判断するアニメーション処理方法。

【請求項10】 請求項3に記載の方法において、

前記対応輪郭線検出工程は、隣接するキーフレームに描かれているオブジェクトの輪郭線どうしの近さを把握した後、対応しあう確率が高い順に輪郭線の対を表示し、ユーザに対応関係を指定する機会を与えるアニメーション処理方法。

【請求項11】 請求項3に記載の方法において、

前記対応輪郭線検出工程は、隣接するキーフレームに描かれているオブジェクトの輪郭線の一方が開曲線、他方が閉曲線の場合、前記閉曲線が前記開曲線の両端をつないだ状態に対応する場合にはユーザがその旨の指定をする機会を与えるアニメーション処理方法。

【請求項12】 フレームをキーフレームとそれ以外の中間フレームに分類して処理を行うアニメーション処理方法において、

オブジェクトが描かれた複数のキーフレームを準備するキーフレーム準備工程と、

アニメーション立体を生成する工程であって、その立体の断面の輪郭線形状を前記キーフレームに描かれたオブジェクトの輪郭線形状に一致させたうえで、その立体の表面を前記断面の輪郭線どうしをつなぐホモトピーの概念を用いて生成する立体生成工程と、

を含むことを特徴とするアニメーション処理方法。

【請求項13】 請求項12に記載の方法において、

前記キーフレーム準備工程は、フレームに描かれるべきオブジェクトの二次元形状が時間の経過とともに位相同形でなくなるようなフレームをキーフレームとして準備するアニメーション処理方法。

【請求項14】 請求項12に記載の方法において、該方法はさらに、

生成されたアニメーション立体の前記断面とは異なる断面における輪郭線形状をもとに中間フレームに含まれるオブジェクトの二次元形状を取得する補間工程を含むことを特徴とするアニメーション処理方法。

【請求項15】 請求項12に記載の方法において、

前記アニメーション立体は、前記キーフレームの数と同じ数の互いに平行な断面によって定義される立体であって、それらの断面に垂直な軸がキーフレームの表示時刻を示す時間軸に対応するような三次元形状をもつ立体であるアニメーション処理方法。

【請求項16】 フレームをキーフレームとそれ以外の中間フレームに分類してアニメーション処理を行うプログラムを格納した記録媒体であって、

そのプログラムは、

オブジェクトが描かれた複数のキーフレームの入力を受け付けるキーフレーム受付モジュールと、

アニメーション立体を生成するモジュールであって、その立体の断面の輪郭線形状を前記キーフレームに描かれたオブジェクトの輪郭線形状に一致させうえて、その立体の位相に注目しながらその立体の表面を生成する立

体生成モジュールと、  
生成されたアニメーション立体の前記断面とは異なる断面における輪郭線形状をもとに中間フレームに含まれるオブジェクトの二次元形状を計算する補間モジュールと、

を含むことを特徴とする記録媒体。

【請求項17】 フレームをキーフレームとそれ以外の中間フレームに分類して処理を行うアニメーション処理システムであって、

オブジェクトが描かれた複数のキーフレームの画像と、  
断面の輪郭線形状が前記キーフレームに描かれたオブ

ジェクトの輪郭線形状に一致するようなアニメーション立体とを送信する送信装置と、

前記画像およびアニメーション立体を受信する受信装置と、  
を含み、前記受信装置は前記アニメーション立体の前記断面とは異なる断面における輪郭線形状をもとに中間フレームに含まれるオブジェクトの二次元形状を取得することを特徴とするアニメーション処理システム。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】 この発明は、アニメーションを処理する方法、特に、画像をフレームをキーフレームとそれ以外の中間フレームに分類して処理を行う方法に関する。この方法は、例えばアニメーション作成の際のインビットウィーニング (inbetweening) またはトゥウィーニング (tweening) と呼ばれる中割処理の自動化に利用することができる。この発明はまた、前記の方法を実現するプログラムを記録する記録媒体、および前記の方法を利用したアニメーションの送受信システムに関する。

【0002】

【従来の技術】 テレビ、ビデオ再生装置をはじめ、各種マルチメディア機器や画像通信技術の進歩により、今日の情報文化、娯楽文化において、映像は音声とともに中心的な役割を果たしている。各種映像に対する需要の拡大が映像技術の進展を促し、映像技術の進歩が新たな映像文化を産んでいる。アニメーションの世界でも、画像の高品質化やコンピュータ・グラフィックス (以下CG) 技術の導入により、かつて想像もできなかった複雑

精緻な、または極めて芸術性の高い作品が製作されるようになった。

【0003】 今日のテレビの標準規格のひとつにNTSCがある。この規格では、表示フレーム数は通常30FPS (frames per second) である。また、リミテッド・アニメーションという簡易的な表示の場合は15FPS、映画の場合はこの数字が通常24FPSとなる。したがって、テレビや映画などのアニメーションを作成するとき、最低でも15FPSに相当するセル面を描く必要がある。1時間のアニメーションの場合、セル画は数万という膨大な枚数になる。

【0004】 こうしたセル画を効率的に制作する手法として、従来より中割手法が知られている。この手法は、まずアニメータがキーフレームと呼ばれる重要なフレームを描き、キーフレーム間のフレーム (以下「中間フレーム」という) を他の数名のアニメータが分業して描き足すというものである。ただし、描画自体は依然手作業によるため、効率化には限界がある。

【0005】 最近では、この中割手法がCGのアニメーション・プログラムにも取り入れられている。こうしたアニメーション・プログラムでは、時間軸上にオブジェクトの動きを割り付けていく。キーフレームはイベントの起こった箇所に設けられる。例えば金槌でくぎをたたくアニメーションを作る場合、キーフレームは2つ存在する。1番目のキーフレームは金槌を振り上げて構える場面、2番目は金槌がくぎに当たる場面である。したがって、まずこれらのキーフレームにおけるオブジェクトの画像をコンピュータに入力する。

【0006】 つづいて、2つのキーフレーム間のオブジェクトの位置を計算することによって中割が行われる。最も簡単な計算は線形補間であり、オブジェクトは2つのキーフレーム間を一定の速度で移動していく。最近では、キーフレーム間でオブジェクトの速度が変わるような場合を想定し、イーズ・イン (加速) やイーズ・アウト (減速) と呼ばれる手法も実現されている。また、オブジェクトの移動経路をスプライン曲線で表現することにより、より自然で滑らかな中間フレームを生成する技術も知られている。

【0007】 特開昭60-191366号公報には、従来一般的な中割技術が開示されている。この公報では、キーフレームに描かれたキャラクタが複数の曲線で構成されるとき、それらの曲線の複数のキーフレーム間における対応関係を予め記述することより、中間フレームにおけるキャラクタの形状を自動的に算出、生成しようというものである。

【0008】

【発明が解決しようとする課題】 各種アニメーション・プログラムに中割機能をもたせることにより、アニメーション制作の効率を改善することができる。しかしながら、中割を行うための準備として、ユーザはキーフレー

10

20

30

40

50

ム間で対応しあう点を予め入力したり、対応しあう点が頂点になるような多角形を描画しなければならないなど、通常のフリーハンドの描画作業とは異なる作業を強いられる。このため、アニメータに負担が生じている。

【0009】さらに大きな問題として、アニメーション・プログラムによって実際に中割を行うと、しばしば中間フレームにおけるオブジェクトが不合理な形状をとることがある。例えば、目の前で掌を $180^\circ$ 回転させるようなアニメーションの場合、キーフレーム間で対応しあう指を自動認識することは容易ではない。したがって、あるキーフレームの中指と次のキーフレームの人差し指の間が間違っ

て対応づけられるなど、現実にはありえない中間フレームが生成されるおそれがある。

【0010】一般に、フレーム間のオブジェクトの正しい対応づけは画像認識の分野で以前から研究されており、エリアマッチング等の種々の手法が提案されている。それにも拘らず、高価なハードウェアと長い計算時間を許したとしても、完全な対応づけはほとんど不可能と考えられている。対応付けの精度を高めるためには、オブジェクトのキーフレーム間における動きが少なくなるよう非常に多数のキーフレームを設ける必要がある。しかし、これではアニメーション制作の効率化に逆行するし、それでも満足できる精度で対応づけがなされることは稀である。特に、掌の回転の具合によって複数の指が重なって見えたり離れて見える場面が混在する場合、画像認識によるアプローチはきわめて困難である。

#### 【0011】

【課題を解決するための手段】本発明の目的は、キーフレームから中間フレームを自動生成する際に中間フレームにおけるオブジェクトの形状が妥当なものになること、準備すべきキーフレームの数を減らすこと、フリーハンドによるキーフレームの描画を認めること、キーフレームに予めフレーム間の対応情報を与える必要をなくすこと、比較的簡素なハードウェアを用いて短い計算時間で中間フレームを生成すること、これら一連の処理に当たってユーザの操作性が簡単であることにある。

【0012】これらの目的を達成するために、本発明者は、まずどのような場合に中間フレームにおけるオブジェクトの形状が不合理になりうるかを解析した。その結果、位相幾何学(トポロジー)の概念を導入することにより、この問題が説明できることを明らかにした。すなわち、オブジェクトの形状の妥当性はそのオブジェクトの位相という観点で捉えることができるのである。本発明者は、位相幾何学、特に後述のモース理論を拡張することにより、オブジェクトの形状の妥当性を確保するための方法を見いだした。

【0013】本発明では、キーフレームにおけるオブジェクトの形状が、ある立体の断面の輪郭線形状に等しくなるような立体を想定する。この立体をアニメーション立体と呼ぶ。いったんアニメーション立体ができれば、

あとはこの立体の断面を切り出すことにより、中間フレームにおけるオブジェクトの形状を得ることができる。逆にいえば、ある断面の形状がその断面の位置に対応する中間フレームにおけるオブジェクトの形状に一致するような立体がアニメーション立体である。

【0014】例えば第1のキーフレームに1つの球A、第2のキーフレームには2つの球B、Cが描かれていたとする。このとき、

ケース1. 球B、Cのうち一方(球Bとする)は球Aそのものであり、他方(球Cとする)はいずれかの中間フレームで出現した、と考えることができる。この他に、ケース2. 球Aが分裂して球B、Cになった、という可能性もある。もちろん、この他にもいくつかの可能性はある。

【0015】しかしいずれの場合でも、アニメーション立体を導入することにより、位相幾何学の観点から説明することが可能である。すなわち、1. は「球AとBは同じ連結成分に含まれ、球Cは別の連結成分に含まれる」と表現できる。連結成分とは、それぞれが個別の実体と考えてよい。つまり、アニメーション立体は球AとBを断面として含む立体と、球Cを断面として含む立体の、合計2つの立体の集合として構成される。一方、2. は「球A、B、Cはすべて同じ連結成分に含まれる」と表現できる。この場合、アニメーション立体は単一の立体で構成される。アニメーション立体は、球Aを断面とする箇所から球BおよびCを断面とする箇所間で分岐する。分岐に関する考察も位相幾何学によって可能になる。

【0016】このように、本発明では位相という概念を意識してアニメーション立体を生成するため、中間フレームにおけるオブジェクトの妥当な形状を条件に応じて提示することができる。例えば、一般則として「球は分裂しない」という条件が与えられれば、本発明では正しく上記1. の場合の中間フレームを生成する。分裂という現象は位相幾何学では分岐(ブランチ)として扱われる。オブジェクトの形状のとりうる可能性の明確な分類は、位相幾何学の助けを借りることによって可能となるのである。

【0017】(1) 具体的には、本発明のアニメーション処理方法は、フレームをキーフレームとそれ以外の中間フレームに分類して処理を行う。この方法は、オブジェクトが描かれた複数のキーフレームを準備するキーフレーム準備工程と、アニメーション立体を生成する工程であって、その立体の断面の輪郭線形状を前記キーフレームに描かれたオブジェクトの輪郭線形状に一致させたうえで、その立体の位相に注目しながらその立体の表面を生成する立体生成工程とを含む。

【0018】この構成において、まず複数のキーフレームが準備される。つづいて、アニメーション立体を生成する。アニメーション立体の形状はその複数の断面の輪

郭線形状によって規定され、それら輪郭線形状はそれぞれ対応するキーフレームに描かれたオブジェクトの輪郭線形状に一致する。つづいて、アニメーション立体の位相に注目しながらその立体の表面を生成し、立体が完成する。

【0019】いったんアニメーション立体が完成すれば、あとはこの断面を切り出すことで任意の中間フレームが得られる。このため、少ないキーフレームから多数の中間フレームを生成することができる。また、アニメーション立体の状態を保存することにより、アニメーションのデータベース化が可能になる。

【0020】(2) 本発明のある態様は、生成されたアニメーション立体の前記断面とは異なる断面における輪郭線形状をもとに中間フレームに含まれるオブジェクトの二次元形状を取得する補間工程を含む。この補間工程がいわゆる中割工程である。

【0021】(3) 本発明のある態様では、前記立体生成工程は、隣接するキーフレームに描かれているオブジェクトの輪郭線どうしの近さをもとに隣接するキーフレーム間における輪郭線の対応関係を把握する対応輪郭線検出工程と、対応する輪郭線間の表面をホモトピーの概念を用いて生成する表面生成工程とを含む。

【0022】例えば前述のケース 1. の場合、球 A が球 B、C のいずれになったかを判定する際、球 A と球 B、球 A と球 C、それぞれの近さを計算する。近さは、例えば球 A の輪郭線上に一定間隔で点を取り、これらの各点から球 B の輪郭線上の最も近い点までの距離を求めてこれらを合計し、この合計値の小ささで評価する。この合計値を球 C についても求め、例えば球 B に関する合計値のほうが小さいければ球 A は球 B になったと考える。この作業を繰り返すことにより、異なるキーフレーム間で対応しあう輪郭線の対（以下「対応輪郭線対」という）が判明する。

【0023】つづいて、対応輪郭線対の間の表面をホモトピーの概念で生成する。ホモトピーの正確な定義は後述するが、概略的には、ホモトピーはある関数  $f$  を別の関数  $g$  に連続的に変形していく関数  $F$  である。したがって、対応輪郭線対の一方の形状関数を  $f$ 、他方のそれを  $g$  としてホモトピー  $F$  を定めれば、その  $F$  によって対応輪郭線対の間の表面が生成される。これをすべての対応輪郭線対の間で行うことにより、アニメーション立体が完成する。

【0024】この態様によれば、異なるキーフレーム間で互いに近い位置にある輪郭線どうしが対応するであろうという経験則をもとに、輪郭線の対応関係の把握を自動化することができる。このため、ユーザはキーフレームをフリーハンドで描画することができ、わざわざキーフレーム間の画像の対応関係を入力する必要もない。中間フレームにおけるオブジェクトの形状も妥当なものになる。

【0025】また、この態様によれば、ホモトピーの概念によって立体の表面が生成されるため、このホモトピーの関数形を決めておけば表面の一意的な表現が可能となる。立体のモデリングの際に表面パッチの貼り付けを行う場合、通常その貼り付け方が一意的には決まらず、したがって自動化が困難とされる。本発明は副次的な効果としてアニメーション立体の一意的な表現が可能になり、したがってアニメーション自体の一意的な表現が可能になる。

【0026】(4) 本発明のある態様では、前記対応輪郭線検出工程は、隣接するキーフレーム間で輪郭線の数一致しないとき、それらのキーフレーム間で輪郭線の自然消滅または自然発生があったものとみなし、その自然消滅または自然発生を考慮して輪郭線の対応関係を把握する。ここで「自然消滅」「自然発生」とは、分岐によって発生または消滅するのではなく、それ自体、他と無関係に発生または消滅することをいう。

【0027】この態様は、上述の球の例においてケース 1 を選択することに等しい。すなわち、ケース 1 では球 A の輪郭線は球 B のそれに連続的につながっていくが、球 C の輪郭線はいずれかの中間フレームで自然発生する。一方、ケース 2 では球 A の輪郭線が球 B および C に分岐するだけで、新たな輪郭線が自然発生するわけではない。

【0028】この態様は「実存する三次元のオブジェクトが分裂したり融合する可能性は低い」という経験則に基づいている。球の例でも、ケース 2 よりはケース 1 のほうがより一般的な現象である。したがって、本態様のような決め方をすることにより、ユーザがケースを指定しなくても、大抵の場合に所望のアニメーションを得ることができる。この結果、ユーザの操作はより簡単になる。

【0029】(5) 本発明のある態様では、前記対応輪郭線検出工程は、隣接するキーフレーム間で輪郭線の自然消滅または自然発生があった場合、輪郭線の木構造をもとに自然消滅または自然発生すべき輪郭線を推定して輪郭線の対応関係を把握する。

【0030】例えば、トーラス（ドーナツ）が  $90^\circ$  回転したとき、最初に中心の穴が見えていたとしても、これが見えなくなる場合がある。輪郭線の中に別の輪郭線があるとき、前者を親輪郭線、後者を子輪郭線と定義すれば、トーラスの外側の円は親輪郭線、内側の円は子輪郭線となる。したがって、消えるのは子輪郭線のほうである。通常の三次元オブジェクトでは、子輪郭線は親輪郭線の存在を前提に存在するため、子輪郭線だけが残って親輪郭線が消えるという現象は起こらない。同様に、自然発生する輪郭線が既存の輪郭線の親輪郭線になることもない。本態様では、こうした観点から自然消滅または自然発生する輪郭線を推定する。「木構造」とは輪郭線の親子関係を示すもので、前記推定を可能ならしめる

ものである。

【0031】本態様によれば、ユーザが自然消滅等する輪郭線をわざわざ指定する必要がなく、ユーザの操作がより容易になる。

【0032】(6) 本発明の別の態様では、前記対応輪郭線検出工程は、アニメーション立体の連結成分の数に関する指定にもとづいて輪郭線の分岐を考慮し、輪郭線の対応関係を把握する。

【0033】球の例において、ケース1で関係する連結成分は2であり、輪郭線の数1から2に変化し、輪郭線は分岐しない。このため、「輪郭線の数に連結成分の数以下のときは輪郭線の分岐はない」と一般化することができる。一方、ケース2で関係する連結成分は1であり、輪郭線数はケース同様に1から2に変化する。ケース2では輪郭線は分岐している。このため、「連結成分よりも輪郭線が多いときは輪郭線の分岐がある」と一般化することができる。

【0034】このように、予め隣接するキーフレーム間で関係する連結成分の数が与えられれば、あとは輪郭線の数から自動的に輪郭線の分岐の有無を判定することができる。

【0035】(7) 本発明のある態様では、前記表面生成工程は、対応する輪郭線上において対応しあう点（以下「対応点对」ともいう）をそれらの点の近さをもとに検出し、これらの対応点对をホモトピーの軌跡で連続的につなぐことで立体の表面を生成する。

【0036】本態様では、対応点对を自動的に決めることができるため、アニメータがキーフレーム間の対応点对をわざわざ指定したり入力する必要がない。

【0037】(8) 本発明のある態様では、(7)のとき前記表面生成工程は、対応する2つの輪郭線A、Bそれぞれの上の点a、bについて、点aから見て輪郭線B上の最も近い点が点bであり、かつ点bから見て輪郭線A上の最も近い点が点aであるとき、これら点a、bを対応点对として検出する。以降、これらの点を「最近点对」ともよぶ。

【0038】この態様は後述の連続トロイダルグラフを用いた対応点对検出の基本原則に当たる。この方法によれば、輪郭線A、B上の対応点对がそれぞれの輪郭線全周を見渡したうえで決まるため、従来の離散トロイダルグラフ（後述）に比べてより自然な対応関係を見い出すことができる。

【0039】(9) 本発明のある態様では、前記対応輪郭線検出工程は、隣接するキーフレームに描かれているオブジェクトの移動による影響を差し引いたうえで、それらキーフレーム上に存在する輪郭線どうしの近さを判断する。

【0040】例えば、カメラでオブジェクトを撮影するような場合、カメラの移動によってすべてのオブジェクトが移動する。この状態でキーフレーム間の対応輪郭線

対を検出すると、誤対応が増えると考えられる。カメラが動かなくとも、複数のオブジェクトが右から左に移動するような場合に同様の問題が起こる。そこでこの態様では、移動の影響を差し引いたうえで輪郭線どうしの近さを判断する。具体的には、すべてのオブジェクトをもとの位置に戻したうえで近さを判定する方法がある。

【0041】この態様によれば、対応輪郭線対の検出精度が高まるため、準備すべきキーフレームの数をさらに減らすことができる。

【0042】(10) 本発明のある態様では、前記対応輪郭線検出工程は、隣接するキーフレームに描かれているオブジェクトの輪郭線どうしの近さを把握した後、対応しあう確率が高い順に輪郭線の対を表示し、ユーザに対応関係を指定する機会を与える。

【0043】この態様によれば、輪郭線対を検出する際に生じうる誤対応の問題を容易に解消することができる。ユーザの指定操作も簡単である。

【0044】(11) 本発明のある態様では、前記対応輪郭線検出工程は、隣接するキーフレームに描かれているオブジェクトの輪郭線の一方が開曲線、他方が閉曲線の場合、前記閉曲線が前記開曲線の両端をつないだ状態に対応する場合にはユーザがその旨の指定をする機会を与える。開曲線とは、有限の長さをもつ曲線であって閉曲線でないもの、すなわち曲線セグメントをいう。

【0045】対応する輪郭線の一方が開曲線、他方が閉曲線の場合、閉曲線が開曲線の両端をつないだ状態に対応する場合のほか、例えば閉じている目が開くように、両端が固定されたまま中央部分が2つに開いて分かれた状態に対応する場合がある。そこで、本発明ではユーザがそれらのいずれを望むかを指定できるものとする。

【0046】(12) 本発明の別の態様は、オブジェクトが描かれた複数のキーフレームを準備するキーフレーム準備工程と、アニメーション立体を生成する工程であって、その立体の断面の輪郭線形状を前記キーフレームに描かれたオブジェクトの輪郭線形状に一致させたうえで、その立体の表面を前記断面の輪郭線どうしをつなぐホモトピーの概念を用いて生成する立体生成工程を含む。

【0047】ここでは(1) 同様にまず複数のキーフレームが準される。つづいて、立体生成工程でアニメーション立体が生成される。アニメーション立体の表面はホモトピーの概念を用いて生成される。この結果、いつでも中割が可能な状態が実現される。

【0048】(13) 本発明のある態様では、(12)のキーフレーム準備工程は、フレームに描かれるべきオブジェクトの二次元形状が時間の経過とともに位相同形でなくなるようなフレーム（以下「臨界フレーム」ともいう）をキーフレームとして準備する。臨界フレームの例は、1つの物体が引き延ばされて2つの物体に分裂す

るとき、その分裂する瞬間のフレームである。本明細書では、アニメーション立体の位相が変わることと臨界フレームが存在することは同義と考えてよい。なぜなら、アニメーション立体の臨界フレームに相当する断面において、1つの物体が2つの物体に分岐等するためである。

【0049】この態様では、実際にオブジェクトの位相が変化する場面でその変化がキーフレームに反映される。したがって、中間フレームでは逆に位相の変化がないことになり、ホモトピーによる表面の生成に好都合である。なぜなら、ホモトピー自体は連続関数であり、ホモトピーの軌跡して生成された表面はアニメーション立体の位相を変化させないためである。

【0050】この態様によれば、キーフレームから中間フレームを生成する際、中間フレームにおけるオブジェクトの形状が妥当なものになる確率が高まる。なぜなら、位相の変化はキーフレームによって指定でき、一方、キーフレーム以外の位相の変化しないフレームは位相を変化させないホモトピーの概念を用いて生成されるためである。この態様では、システムが位相の変化を推定する必要はない。実際のアニメーション作成現場でも、アニメータは無意識のうちに臨界フレームをキーフレームにしていることが多く、インプリメンテーションの面でも無理がない。

【0051】(14) (12)において本発明のある態様は、生成されたアニメーション立体の前記断面とは異なる断面における輪郭線形状をもとに中間フレームに含まれるオブジェクトの二次元形状を取得する補間工程を含む。このため、(2)同様、容易に中割が実現される。

【0052】(15) (12)において本発明のある態様では、前記アニメーション立体は、前記キーフレームの数と同じ数の互いに平行な断面によって定義される立体であって、それらの断面に垂直な軸がキーフレームの表示時刻を示す時間軸に対応するような三次元形状をもつ立体である。すなわち、アニメーション立体の断面を $xy$ 平面に平行におくとき、 $z$ 軸が時間軸となる。

【0053】仮に2つのキーフレームの表示時刻を $t = t_0$ 、 $t = t_1$ とすると、これら2つのキーフレームのちょうど中間にある中間フレームの表示時刻は $t = (t_0 + t_1) / 2$ である。2つのキーフレームはそれぞれアニメーション立体の $z = t_0$ 、 $z = t_1$ における断面である。また、前記中間フレームは同様に $z = (t_0 + t_1) / 2$ における断面となる。アニメーション立体をこのように定義することにより、中割処理を少ない計算量で容易に行うことができる。

【0054】(16) 一方、本発明のさらに別の態様は記録媒体に関する。この記憶媒体にはアニメーション処理を行うプログラムが格納される。このプログラムは、オブジェクトが描かれた複数のキーフレームの入力を受

け付けるキーフレーム受付モジュールと、アニメーション立体を生成するモジュールであって、その立体の断面の輪郭線形状を前記キーフレームに描かれたオブジェクトの輪郭線形状に一致させたうえで、その立体の位相に注目しながらその立体の表面を生成する立体生成モジュールと、生成されたアニメーション立体の前記断面とは異なる断面における輪郭線形状をもとに中間フレームに含まれるオブジェクトの二次元形状を計算する補間モジュールとを含む。このプログラムは(1)および(2)の処理を行うものと考えられる。

【0055】(17) 本発明のさらに別の態様はアニメーションを通信するシステムに関する。このシステムは、オブジェクトが描かれた複数のキーフレームの画像と、断面の輪郭線形状が前記キーフレームに描かれたオブジェクトの輪郭線形状に一致するようなアニメーション立体とを送信する送信装置と、前記画像およびアニメーション立体を受信する受信装置とを含む。

【0056】この構成において、受信装置がアニメーション立体を受信した後、アニメーション立体の断面を切り出すことにより、その断面における輪郭線形状をもとに中間フレームに含まれるオブジェクトの二次元形状を取得する。ある程度多数の中間フレームを取得すれば、これらを順次表示することにより、アニメーションを再生することができる。

【0057】このシステムによれば、非常に少ない通信データ量でアニメーションを送ることができる。

【0058】

【発明の実施の形態】本発明の好適な実施形態を説明する。本発明を理解するに際し、本発明者が先に公表した論文の内容を「前提技術」として説明することは有用である。この前提技術は本発明者のひとりの論文(東京大学博士論文1993年品川嘉久)の一部である。

【0059】前提技術は、位相幾何学の考え方を導入して立体を正確に表現するための技術である。一方、本発明は立体の三次元形状ではなく、その断面の二次元形状を最終目的とし、立体を異なる意味合いで用いる。本発明では、立体の断面の形状からアニメーションの各フレーム画像を得る。以下、前提技術を引用した後、その前提技術に対して必要な修正または拡張を加える形で実施形態を説明する。

【0060】[前提技術]

[1] モース理論に基づく曲面符号化システム

(1) はじめに

モース(Morse)理論を数学的ツールとして三次元物体の解釈を行う。しかし、三次元の曲面を完全な正確さをもって符号化するためにはモース理論だけでは不十分である。以下、その理由を説明し、モース理論を拡張することによってこの問題の解消を図る。

【0061】三次元空間における立体や曲面の形状を符号化するとき、通常これらをなんらかの記号の配列とし

て表現する。自然界に見られるオブジェクトの場合は、形状は非常に多くの自由度をもつ。そのため符号化の際には一定の単純化が必要となる。位相幾何学（トポロジー）はこうした単純化を行うための数学的手段である。

【0062】（2）符号化システムに対する要請  
自然界の物体の形状データは、しばしば断面という形で利用可能である。例えばX線断層撮影などである。このデータを符号化に用いれば、隣接する断面の輪郭線間で三角パッチなどの内挿を行う曲面再構成システムによる利用が可能になる。以下説明する符号化方法も断面の輪郭線（以下単に「輪郭線」とも呼ぶ）に基づく。

【0063】符号化システムは、例えば人の内耳の半規管のように、分岐、ハンドル（穴）または多層曲面からなる内部構造をもつような物体も符号化できなければならない。こうした物体を符号化するために、一連の断面の輪郭線を記録していくだけでは、対象である曲面を十分に説明することにはならない。

【0064】符号化システムでは、符号化された曲面の位相的な正しさ（トポロジカル・インテグリティ）が維持される必要がある。そうでないと、符号化された曲面が物理的に意味をなさないことがある。図1は輪郭線構造の不当な変換例を示している。ここでは2つの輪郭線があり、一方が他方に包含されている。したがって、本来前者は後者の境界線を超えることはできず、この規則が守られない限り正しい符号化はできない。曲面符号化システムには位相的な正しさの維持が要求されるのである。

【0065】（3）古典的なモース理論  
もともとモース理論は変分法を取扱うために提唱された。そしてその目的は、無限次元の道の空間における汎関数の極小値を記述することにあつた。このことから逆に、汎関数の極小値を利用することにより、それ以外の方法での記述が困難であるような空間の位相的な特徴を記述することが可能になる。以下、モース理論の概要を説明する。

【0066】◎ 可微分多様体  
モース理論を適用できる空間は可微分多様体である。有限次元の多様体を考えてみる。

【0067】いま任意の整数 $n$ について、 $n$ 次元多様体は位相空間であつて、そこではすべての点が $n$ 次元空間 $R^n$ の部分集合の上に一对一かつ両連続に写像可能な近傍をもつとする。このような写像は「チャート」と呼ばれ、その領域に含まれる点について局所座標系を提供する。地球を例にとれば、緯度と経度が局所座標系に当たる。多様体が $p$ 回微分可能であるためには、一方の座標系から他方の座標系への変換が、2つの異なるチャートの値域に含まれる点について $p$ 回微分可能でなければならない。

【0068】このため多様体は、可微分的に重なり合った $R^n$ の領域から構成されていると考えることができ

る。例えば、直線や円周は一次元多様体の構造を与えられる。また、球の表面は例えば北半球と南半球のように、少なくとも2つのチャートを用いた二次元多様体を用いて表現できる。同様にトーラスの表面は、少なくとも4つのチャートを用いた二次元多様体によって表現できる。 $R^3$ から結び目のある円を取り除けば三次元多様体の一例となるし、さらに高次の多様体は例えばロボットの腕のコンフィギュレーション空間として現れる。

【0069】◎ 可微分写像および特異点

チャートを用いることにより、 $p$ 次元多様体から $n$ 次元多様体への写像は $R^p$ の各区分から $R^n$ の各区分への写像として（区分ごとに）数値表現することができる。これらの写像については微分可能性を検証することができる。要素が $k$ 回連続的に微分可能であれば、その写像は $C^k$ 級である。

【0070】ここで局所座標系の上で高さ関数を定義する。高さ関数は与えられた点の高さ（物体が埋め込まれている三次元空間における $z$ 座標など）を返す関数である。

高さ関数 $h: R^2 \rightarrow R$ のヤコビ行列は、

【数1】

$$J = \begin{pmatrix} \frac{\partial h}{\partial x_1} \\ \frac{\partial h}{\partial x_2} \end{pmatrix}$$

で与えられる。ヤコビ行列は各点について計算することができ、そのランクの最大値は $n$ と $p$ の小さいほうである。ヤコビ行列のランクがこの最大値に等しい点は「正則点」と呼ばれ、それ以外の点は「特異点（singular point）」または「臨界点（critical point）」と呼ばれる。例えば、高さ関数に関する特異点には、頂上（peak）、鞍点（saddle point）、谷底点（pit）がある。別のいいかたをすれば、特異点はヤコビ行列がゼロベクトルとなる点であり、特異点では法線ベクトルが高さ方向と同じ方向を向く。なお、特異点が写像される点は特異値と呼ばれる。

【0071】◎ ヘッセ行列と指数

$n$ 次元多様体から $R$ への写像（以下「多様体上の関数」と呼ぶ）については、ある点における偏微分の値がすべて0の場合、その点が臨界点となる。そのような点において、前述の関数は二次偏微分に基づく二次形式によって近似される。この行列表示はヘッセ行列と呼ばれ、その要素は以下のように記述される。

【0072】

【数2】



$$h = \begin{pmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_1 \partial x_2} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{pmatrix}$$

特異点におけるヘッセ行列の負の固有値の数をその特異点の指数 (index) と呼ぶ。指数は後述の図 2 に示すとおり、被約形式におけるマイナス符号の数に等しい。すなわち、頂上の指数は 2、鞍点は 1、谷底点は 0 である。後述の図 4 (a) ~ (c) のごとく、トーラスの場合、指数 2、1、0 の臨界点はそれぞれ 1 個、2 個、1 個である。

【0073】臨界点におけるヘッセ行列のランクが  $n$  であるとき、その特異点は縮退していない、すなわち「非縮退」と呼ばれる。どのような  $C^2$  級関数でも、モース関数によって近似することができる。モース関数とは、臨界点の縮退がないような関数をいう。したがって、モース関数の臨界点は孤立しているはずであり、コンパクトな多様体に関する限り、臨界点は有限個しか存在しない。いずれの 2 つの特異値も等しくないと仮定することは非常に都合であり、その仮定の下でも同じ近似結果が得られた。

#### 【0074】◎ ホモトピーの型

モース理論によれば、多様体とその多様体上のモース関数が与えられ、その関数の特異点の指数の列び方が判明すれば、各特異点に対応する一連の演算を行うことによって、その多様体と同じホモトピー型をもつ位相空間を胞複体 (cell complex) として構築することができる。

【0075】高さを表す任意の実数について、セル (胞体) はその実数の示す高さ以下の点からなる多様体の部分に関するモデルを与える。R が上下に走査されるとき、2 つの連続する特異値の間ではセルの位相は変化しないが、特異値を横切るたびに、それ以前のセルに対して「 $k$  次元セル」をつなげていくことにより、胞セルを作り上げていくことができる。ここで  $k$  は横切った特異点の指数である。簡単にいえば、物体の形はその臨界点の指数と同じ次元をもつセルというものを貼り合わせることで復元できる。

【0076】図 2 は、特異点の指数、 $k$  次元セルおよびそれによって符号化される物体の関係を示す図である。ここでは物体としてトーラスを挙げている。同図のごとく、注目する高さが臨界点を含む高さを横切るとき、その高さより下の点によって構成される位相が変化する。この変化は、位相的に見れば後述するように  $k$  次元セルをつなげていくことと同じである。同図のごとく、二次元セル ( $k=2$ ) はお椀を伏せたような形、一次元セル ( $k=1$ ) は紐のような形、0 次元セル ( $k=0$ ) はひとつの点で表すことができる。もとの物体の形は、それらのセルをつなげた上で、粘土細工のように変形するこ

とで得られる。トーラスの場合、二次元セルを 1 個、一次元セルを 2 個、0 次元セルを 1 個つなげて得られる。

【0077】ここで注意すべきは、指数の配列だけではセルを完全に記述することはできないことである。図 3 (a) ~ (c) は、それぞれが同じモースの指数の配列をもつ 3 組の曲面を示す。このように、指数の配列だけでセルを完全に決めることはできない。そのため、セルをつないでいくときにいずれの連結成分 (それぞれ独立した実体) が関連するかを知らなければならない。

【0078】レーブ (G. Reeb) は、多様体から位相商空間として得られるグラフを提唱した。レーブグラフは特異点の相互関係を示すもので、物体表面を等高線で表し、各等高線の連結成分をひとつの点として表すことで得られる。レーブは、多様体 (コンパクトとする) において、モース関数の下で同じ値をもち、かつ対応する断面として同じ連結成分に含まれるすべての点を等化することにより、このグラフを導出した。つまり、2 つの臨界点を含む平面間に存在する多様体の部分の連結成分はグラフの辺として表現され、各特異点はグラフの各頂点に対応する。レーブグラフは物体の骨格を示すグラフとすることができる。

【0079】図 4 (a) ~ (c) は、トーラスとそのレーブグラフの関係を示す図である。図 4 (a) は、もとのトーラス、図 4 (b) はその断面図、図 4 (c) はレーブグラフを示している。図 4 (b) において、同一平面内にあって重なり合わない円の部分が、(c) における 2 つの別々の辺に相当する。このレーブグラフはアイコンとして極めて表現力に優れているため、以降必要に応じてこのグラフをアイコン表示に用いる。

#### 【0080】(4) 理論上の限界

重要なのは、モース理論をこのように古典的な方法で用いた場合、多様体に内在する位相的な性質を発見することができるに過ぎないことである。指数の配列だけでは、多様体が空間に埋め込まれている状態を符号化することができない。例えば、空間に埋め込まれたトーラスに結び目があるかどうかは知ることができない。図 4 (b) に示すとおり、2 つの異なる形状が同じ特異点に帰着するためである。同様に図 4 (c) に示すとおり、連結があるかどうかともモース理論による単純な符号化では示せない。

【0081】(5) モース理論に基づく符号化の拡張  
ここでは、議論の対象を  $C^2$  級 (三次元空間に埋め込まれた  $C^2$ ) のコンパクトな二次元多様体の表面に限る。我々が用いる曲面上のモース関数は、空間における高さ関数から誘導する。事実、 $C^2$  曲面をわずかに回転させれば臨界点の縮退をなくすることができるため、その高さ関数をモース関数にすることができる。

【0082】モース理論によれば、2 つの臨界レベル (臨界点が含まれる平面の高さ) の間では断面の位相は変化しない。このことから、曲り具合の異なる多くの円

筒を用いて、2つの臨界レベルの間の曲面をモデル化することができる。特異点のない断面は平面に埋め込まれた円から構成されるため、包含関係を表示するために構造的な符号化が必要となる。すなわち同じ円に含まれる複数の円をグループ化する符号化である。

【0083】我々は、レーブグラフから得られる情報（すなわち各頂点間の接続状態）の他に、モースの指数に新たな情報を付け加える拡張符号化を提案する。すなわちこの情報は、2つの連続する臨界値の間で複数の円筒がどのように交換され、どのような向きに接続されるかに関する情報である。

【0084】(6) 符号化システムの例  
符号化システムの概要を説明する。このシステムでは、曲面に対して $k$ 次元セルを次々につないでいくことにより曲面を表現する。そして各断面における輪郭線の階層構造の変化を追跡する。ここで、セルの接続を示す演算子を導入し、これらの演算子を用いて曲面を符号化する。演算子によって接続されていくセルをアイコンで表現することにより、符号化の対象となる曲面の構造の理解を容易にする。

【0085】この符号化システムの最大の特徴は、得られる符号化の結果が位相の正しさを保証することにある。図1に示したとおり、断面間における輪郭線の階層構造の変化が物理的に意味のある曲面を規定するとは限らない。つまり、各断面における輪郭線の階層構造を符号化するだけでは、位相の正しさを保証することができない。したがって、輪郭線の階層構造を変化させるような演算を符号化することが必要になる。

【0086】◎ 親子関係と輪郭線  
輪郭線の階層構造の表現方法を説明する。ここでは木構造を用いる。

【0087】まず、ある輪郭線が別の輪郭線を包含するとき、前者を後者の親輪郭線、後者を前者の子輪郭線と呼ぶ。図5(a)と(b)は輪郭線の親子関係、およびその木構造による表現を示す図である。図5(a)のごとく、親子関係はネスティング構造にすることができる。以下、輪郭線の一番目を #1 で示す。#1 は #2 の親輪郭線、#2 は #4 の親輪郭線である。#1 や #7 のように親輪郭線をもたない輪郭線は「仮想輪郭線 #0」の子輪郭線と表現する。したがって、#0 は木構造の頂点にくる。

【0088】ある輪郭線の親を示すために、配列  $Parent[i]$  を定義する。たとえば  $Parent[1]=0$  である。一方、ある輪郭線の子輪郭線は  $Children$  という配列にリストされ、その配列へのポインタが付される。例えば #3 の子輪郭線である #5 および #6 は、以下のように記述される。

【0089】  
 $Children[3][1]=5$ ,  $Children[3][2]=6$   
同じ親輪郭線をもつ輪郭線同士は兄弟輪郭線と呼ばれ

る。図5(a)の場合、#2 と #3 が兄弟輪郭線である。親輪郭線の親は祖父輪郭線、子輪郭線の子は孫輪郭線と呼ばれる。また、内部に物体が存在する輪郭線を中実輪郭線、存在しない輪郭線を中空輪郭線と呼ぶ。図5(a)の場合、#1、#4、#5、#6、#7 は中実輪郭線、#2 および #3 は中空輪郭線である。

【0090】◎ セルを接続するための演算子  
4つの演算子、 $Put\_e0$ 、 $Put\_el\_merge$ 、 $Put\_el\_divide$ 、 $Put\_e2$  を定義する。これらがセルを貼り付ける演算子である。以下、 $k$ 次元セルを  $e^k$  と表示する。

【0091】物体の構成は頂上から谷底に向けて進む。処理は、それ以上セルを接続することができなくなった時点で終了する。演算子によって構成しようとする曲面の状態を示すために、各断面における輪郭線を用いる。

【0092】図6は、これらの演算子を用いてトーラスを構成する方法を示す。以下この図を用いて演算子の機能を説明する。

【0093】1. 同図の一番上に示すとおり、 $e^2$  を生成するために  $Put\_e2(0)$  を実行する。このパラメータ「0」は #1 が #0 の内側に生成されることを示す。このセルの断面は同図の「断面表示」の箇所に示される。図のように、 $Put\_e2$  は断面の平面上に輪郭線を生成する機能をもつ。演算子によって生成される輪郭線に生成順の数字を与えるため、 $Put\_e2(0)$  によって生成された輪郭線は #1 である。

【0094】新たに生成された輪郭線の状態は、初期値として常に「イネーブル」である。イネーブルとは、その輪郭線に対してセルを接続することが許される状態を示す。 $e^2$  のアイコン表示を同図「アイコン」の下に示す。

【0095】2. つづいて、 $Put\_el\_divide(1, nil, inside)$  により、 $e^2$  に対して  $e^1$  を貼り付ける。新たに生成された輪郭線を #2 とする。パラメータ「inside」は #2 が  $Parent[1]=0$  の子輪郭線として生成されることを示す。二番目のパラメータは参照すべき子輪郭線のリストを示す。ここでは2番目のパラメータが「nil」であり、ここでは子輪郭線に対する操作、具体的には子輪郭線の削除はない。

【0096】3. つぎに、 $Put\_el\_merge(1, 2)$  を用いて別の  $e^1$  を貼り付け、#1 と #2 をマージする。この演算子は、1番目、2番目のパラメータによって示される輪郭線を1番目のパラメータの側にマージする。マージによって2番目のパラメータの示す輪郭線は消滅するため、それがその親輪郭線のもつ子輪郭線のリストから削除される。同時に、その輪郭線の状態が「イネーブル」から「ディセーブル」に変更される。したがって、この輪郭線に新たにセルを接続することができない。

【0097】4. 最後に  $Put\_e0(1)$  を用いて  $e^0$  を貼り付け、#1 を閉じる。#1 の状態はイネーブルからディセーブルに変更される。アイコンがこの変更を反映して

いる。ある輪郭線にセル  $e^0$  が接続されたとき、その輪郭線のもつすべての子輪郭線が予めディセーブルされていなければならない。

【0098】以上の手順により、イネーブルの状態に残っている輪郭線がなくなるため、演算子によるセルの貼り付けは完了する。

【0099】図7～9は疑似パスカルコードによって演算子のプログラミング例を示す図である。図において、Max\_children と Max\_contour\_number は十分大きな正の整数であり、メモリアロケーションのためだけに用

【0100】図7において、Number\_of\_children は各輪郭線の子輪郭線の数、Most\_recently\_created# は、最も最近生成された輪郭線の番号、Contour\_status は、各輪郭線がイネーブルであるかディセーブルであることを示す。Children はある輪郭線の子輪郭線のリスト

```
procedure create_new_contour;
begin
    most_recently_created#:=most_recently_created#+1;
    contour_status[most_recently_created#]:=enabled;
end
```

図8において、後に使用するために2つのプロシージャと3つの関数を定義する。Add\_listed\_children は第2パラメータである clist にリストされている輪郭線を第1パラメータの子輪郭線 (children [n] ↑) のリストに追加する。Remove\_listed\_children は第1パラメータの子輪郭線 (children [n] ↑) のリストから第2パラメータである clist にリストされている輪郭線を削除する。これらのプロシージャはまた、Number\_of\_children および Parent# の配列を更新する。

【0103】一方、関数 Add\_children (n, clist) は clist のすべての輪郭線が #n の子輪郭線であるとき「真」を返す。それ以外の場合「偽」を返す。関数 in\_list (n, clist) は clist が #n を含むとき「真」を返し、それ以外の場合「偽」を返す。関数 List\_containing\_only (n) は2つプロシージャ、Add\_listed\_children および Remove\_listed\_children に対して与えるべき輪郭線を1つだけ含むリストを作るために定義される。

【0104】これらにより、4つの演算子 Put\_e2、Put\_e0、Put\_e1\_divide および Put\_e1\_merge を定義することができる。これらは図9に示される。

【0105】1. Put\_e2 (n) は #n の子輪郭線として新たな輪郭線を生成する。

【0106】2. Put\_e0 (n) は  $e^0$  を貼り付けることにより、#n を削除する。ここで、All\_successors\_disabled (n, contour\_number) は、#n のすべての子輪郭線がディセーブルであるときに限り「真」を返す。

【0107】3. Put\_e1\_divide (n, clist, inside) は、#n を分割して新たな輪郭線を作る。Clist にリス

を保持する Child\_list という配列に対するポインタである。Child\_list の終了は定数 End\_of\_list によって示される。これらの変数は以下の手順で初期化される。

```
【0101】
most_recently_created#:=0;
for i:=0 to max_contour_number do
    number_of_children[i]:=0;
for i:=1 to max_contour_number do
    contour_status[i]:=disabled;
contour_status[0]:=enabled;
```

これにより、仮想輪郭線がイネーブルされる。つぎの create\_new\_contour は新しい輪郭線を作り、Most\_recently\_created# を増加させ、その状態を初期化する。

【0102】

トされている輪郭線は新たに生成される輪郭線の子輪郭線になる。それらは、#r=#n または #r=parent#[n] の一方が成り立つとき、#r の子輪郭線のリストから削除される。(これは #n または parent#[n] のいずれの子輪郭線であったかに依存する)。これら2つの場合のみが許される。

【0108】4. Put\_e1\_merge (c1,c2) は #c2 を #c1 にマージする。#c2 は Parent#[c2] の子輪郭線のリストから削除される。#c2 のすべての子輪郭線は Parent#[c1] または #c1 の子輪郭線になる (そのいずれであるかは、#c1 が #c2 の親輪郭線であるか兄弟輪郭線であるかに依存する)。これら2つの場合のみが許される。

【0109】これらの演算子によって符号化される曲面の構造を容易に理解するために、曲面のレーブグラフを構成するセルの視覚的表示を提案する。

【0110】図10に示すアイコンはそれぞれセルを表している。セルの貼り合わせは、一方のセルの平らな頂面と他方のセルの平らな底面を接触させることによって行われる。中空輪郭線には白いアイコン、中実輪郭線には黒いアイコンを用いる。 $e^1$  に関しては複数のアイコンが存在する。

【0111】図11はセルの貼り合わせを示す図であり、同図に示すごとく子輪郭線のアイコンはその親輪郭線のアイコンの内側に描かれる。鉛直な軸に関する鏡像のアイコンも認められる。

【0112】アイコンの特徴は、それが貼り付けられる輪郭線の構造を維持する点にある。例えば図11の上半分に示されるように、 $e^1$  がセルに接続されるとき、アイコンが接続されることになるセルの親子関係の構造が

維持される。

【0113】図11の下半分に示すように、セルの高さの調整が必要な場合にはダミーアイコンを挿入する。レーブグラフは平面グラフではないため、ダミーアイコンを用いてセルを交差させることもできる。この結果、 $e^1$ を離れたセルに接続することが可能となる。 $e^1$ がセルを交換するとき、同時にその内部構造も交換する。階層的な輪郭線の構造を保持するために、ダミーアイコンはそれが貼り付けられる輪郭線の親輪郭線の境界を越えることはできないし、他の輪郭線に侵入することもできない。この理由から、ダミーアイコンを用いて交換することのできる輪郭線は兄弟輪郭線に限られる。

【0114】以上のまとめとして、図12～14に演算子を用いた物体の符号化の例を示す。図12は、対象となる物体のレーブグラフをアイコンによって示したものの、図13はその物体の断面の輪郭線を示したものの、図14はその物体を構成するための演算子である。

【0115】[2] 演算子を用いた符号からの曲面の構成

(1) ホモトピーの軌跡としての曲面の生成  
前述の方法によって得られた符号化データをもとに物体の曲面を構成する。

【0116】すでに述べたとおり、臨界点を含む断面どうしの間では輪郭線の位相は変化しない。頂点から底辺まで走査したとき、輪郭線の形状は変化する。この輪郭線の変形はホモトピーを用いてうまく表現することができる。ホモトピーはある関数を他の関数に変換する。以下の説明においては、すべての輪郭線は形状関数によって表され、変形はホモトピーによって表されるとする。ホモトピーの定義は以下のとおりである。

【0117】[定義]  $X, Y$  が位相空間であるとき、 $f, g: X \rightarrow Y$  という写像を考える。ここで、 $x \in X$  なるすべての点  $x$  に対して、

$$F(x, 0) = f(x)$$

$$F(x, 1) = g(x)$$

が成り立つような写像  $F: X \times I \rightarrow Y$  が存在する場合、「 $f$  と  $g$  はホモトープである」といわれる。ここで  $I = [0, 1] \in \mathbb{R}$  である。またこのとき、写像  $F$  は「 $f$  から  $g$  へのホモトピー」と呼ばれる。 $F$  が、

$$F(x, t) = (1-t)f(x) + tg(x)$$

で定義されるとき、これは直線ホモトピーと呼ばれる。図15には輪郭線のホモトピー変形が示されている。この図において、一番上の輪郭線が形状関数  $f$ 、一方いちばん下の輪郭線は  $g$  によって表されている。曲面は  $f$  から  $g$  へのホモトピー  $F$  の軌跡として生成される。

【0118】(2) 演算子をインプリメントするための

$$N_{i,0} = \begin{cases} 1 & (x_i \leq u < x_{i+1}) \\ 0 & \text{otherwise} \end{cases}$$

$$N_{i,k}(u) = \frac{(u - x_i)N_{i,k-1}(u)}{x_{i+k} - x_i} + \frac{(x_{i+k+1} - u)N_{i+1,k-1}(u)}{x_{i+k+1} - x_{i+1}} \quad (2 \leq k \leq n+1).$$

要素

図15の曲面を生成するための演算子は、ホモトピーによって輪郭線を変形するものとして記述することができる。

【0119】1. 演算子を構成する要素

図16には、演算子を構成する以下の主な4つの要素が描かれている。

【0120】(i)  $f: I \rightarrow \mathbb{R}^3$  上の輪郭線の形状を与える

(ii)  $g: I \rightarrow \mathbb{R}^3$  下の輪郭線の形状を与える

(iii)  $F: f$  から  $g$  へのホモトピー

(iv)  $h: 2$  つの輪郭線の高さの差

2. 形状関数

$f, g$  として以下の形状関数を準備する。

【0121】(i) 点 : 常に固定点の位置を与える定数関数

(ii) 円 : 円の形状を与える

(iii) 多角形: 任意の頂点を結ぶ多角形の形状を与える

(iv) ベジエ:  $n$  次元のベジエ曲線で、次式で記述される

【数3】

$$f(u) = \sum_{i=0}^n B_i^n(u) P_i \quad (0 \leq t \leq 1)$$

この関数は制御点  $P_i \in \mathbb{R}^3$  と呼ばれる  $n$  個の点の集合 (順序つき) によって特定される。この制御点はユーザーによって修正することができる。ここで、 $B_i^n(t)$  はベルンシュタインの基底関数であり、次式で定義される。

【0122】

【数4】

$$B_i^n(u) = \binom{n}{i} u^i (1-u)^{n-i}.$$

(v) NURBS (Non Uniform Rational B-Spline) 曲線: この曲線の制御点もユーザーによって定義される。NURBS 曲線は次の式で定義される。

【0123】

【数5】

$$f(u) = \frac{\sum_{i=0}^n N_{i,k}(u) w_i P_i}{\sum_{i=0}^n N_{i,k}(u) w_i}$$

ここで  $w_i$  は各制御点の重みである。 $N_{i,k}(t)$  は B スプライン基底関数と呼ばれる  $(k-1)$  次の多項式の各区分の値を示す。これは次式で定義される。

【0124】

【数6】

NURBSは、非常に多くのCADシステムで用いられている。NURBSは二次曲面を正確に表現することができ、また局所的な近似特性をもっている。すなわち、制御点またはそれに関連する重みが増減したとき、その点の近傍でしか曲面の形状に影響を与えない。

#### 【0125】3. ホモトピーF

$$\text{quadrant: } F(x, t) = \text{quadrant}(x, t) = \sqrt{1-t^2}f(x) + (1-\sqrt{1-t^2})g(x)$$

(iii) 放物線:  $F(x, t) = (1-t^2)f(x) + t^2g(x)$

(iv) カージナルスプライン: カージナルスプラインは、一番上および一番下の輪郭線を内挿補間する。輪郭線間の対応点を示すトロイダルグラフを使うことによってパラメータの決定を自動化することができる。

【0127】(v) ガイディングカーブ: 輪郭線上の点をガイディングカーブに沿って動かすことにより、輪郭線を変形することができる。輪郭線に対して複数のガイディングカーブを付けることができる。輪郭線がベジェ曲線またはNURBS曲線で表されるとき、ガイディングカーブは制御点に付けられ、変形は制御点の動きによって決定される。ガイディングカーブが付けられていない制御点の動きは、隣接する制御点のガイディングカーブを用いて計算することができる。

【0128】図17は、ユーザーがガイディングカーブを付けることにより、上の輪郭線が徐々に下の輪郭線に変形される様子を示している。

【0129】いずれの場合も、結果的に得られた輪郭線の間をカージナルスプラインを用いてパッチを当て、曲面を生成する。

#### 【0130】4. 演算子のための輪郭線の形状関数

演算子のための輪郭線の形状関数は以下のように与えられる。

【0131】(i) Put\_e2

f: 点

g: ユーザーが特定 (デフォルト: 円)

F: ユーザーが特定 (デフォルト: 四分円形)

(ii) Put\_e0

f:  $e^0$  が付けられる輪郭線の形状関数

$$c(t) = \begin{pmatrix} (1-\sqrt{1-(2-2t)^2})c(1/2)_x + \sqrt{1-(2-2t)^2}c(1)_x \\ (1-\sqrt{1-(2-2t)^2})c(1/2)_y + \sqrt{1-(2-2t)^2}c(1)_y \\ (2-2t)c(1/2)_x + (2t-1)c(1) \end{pmatrix}$$

与えられる。この道の初期値のxy平面への射影は、 $c(0)$  および  $c(1)$  を結ぶ線分になる。この  $c(t)$  の式において、ルート  $(1/2)$  の部分を  $(1-x^2)$  に置き換えれば放物線の道を得ることもできる。Put\_el\_divide および Put\_el\_merge の要素は以下のとおりである。

【0132】Put\_el\_divide

f:  $e^1$  が付けられる輪郭線の形状関数

c:  $e^1$

$c(0), c(1): c(0)=f(s_1), c(1)=f(s_2)$  なる  $s_1, s_2 \in [0, 1]$  で特定される

一方、ホモトピーFとして以下の関数が導入される。これらの関数は断面の輪郭線を出力する。

【0126】(i) 線形: 直線ホモトピー

(ii) 四分円形:

【数7】

g: 点

F: ユーザーが指定 (デフォルト: 四分円形)

(iii) Put\_el\_divide, Put\_el\_merge

$e^1$  の道  $c: [0, 1] \rightarrow R^3$  を決めなければならない。実際にインプリメントする場合、道  $c$  は  $c(0), c(1/2)$  および  $c(1)$  の位置によって特定される。道は滑らかでなければならない、また  $c(1/2)$  における接線のベクトルは  $xy$  平面に平行でなければならない。したがって、 $c(1/2)$  は生成された曲面の鞍点になる。 $c(1/2)$  の初期位置は、

【数8】

$$c(1/2) = \begin{pmatrix} \frac{c(0)_x + c(1)_x}{2} \\ \frac{c(0)_y + c(1)_y}{2} \\ c(0)_x - h \end{pmatrix}$$

である。ここで、

【数9】

$$c(t) = \begin{pmatrix} c(t)_x \\ c(t)_y \\ c(t)_z \end{pmatrix}$$

である。道の初期値は  $c(0)$  と  $c(1)$  を接続する楕円の弧であり、 $0 \leq t \leq 1/2$  については、

【数10】

$$c(t) = \begin{pmatrix} (1-\sqrt{1-(2t)^2})c(0)_x + \sqrt{1-(2t)^2}c(1/2)_x \\ (1-\sqrt{1-(2t)^2})c(0)_y + \sqrt{1-(2t)^2}c(1/2)_y \\ (1-2t)c(0)_x + 2tc(1/2)_x \end{pmatrix}$$

与えられ、一方、 $1/2 \leq t \leq 1$  については、

【数11】

$s_2 \in [0, 1]$  で特定される

$g_1, g_2$ : 輪郭線を道  $c$  に沿って分割することにより得られる

Put\_el\_merge

c:  $e^1$  の道

$f_1, f_2$ :  $e^1$  が付けられる輪郭線の形状関数

$c(0), c(1): c(0)=f_1(s_1), c(1)=f_2(s_2)$  なる

$s_1, s_2 \in [0, 1]$  で特定される

g: 道  $c$  に沿って輪郭線をマージすることにより得られる

Put\_el\_divide による変形は、 $e^1$ の道をガイディングカーブとして用いることにより、輪郭線を変形することで行われる。すなわち、

$$\cdot F(s_1, t) = c(t/2)$$

$$\cdot F(s_2, t) = c(1-t/2)$$

である。一方、Put\_el\_merge の変形は次式によって得られる。

$$【0133】 \cdot F_1(s_1, t) = c(t/2)$$

$$\cdot F_2(s_2, t) = c(1-t/2)$$

(iv) ダミーアイコン

すでに述べたダミーアイコンを用いることにより、輪郭線の構造を変えることなく輪郭線の形状を変形することができる。ダミー演算子の形状関数は以下のとおりである。

【0134】  $f$  : ダミーが付けられる輪郭線の形状関数  
 $g$  : ユーザが指定 (デフォルト : 円)

このシステムは、子輪郭線が親輪郭線の境界を超えないことを自動的にチェックし、位相上の正しさを維持するものとする。

【0135】 (3) 微分不可能な場合および縮退への対応

これまで、物体表面の曲面が $C^2$ 級の可微分性をもつと仮定してきた。またすべての臨界点是非縮退であると仮定した。しかしながら、円筒や立方体など物体を設計する場合、多面体や、頂点または底辺が平面であるような微分不可能な点または縮退した点を含めて符号化できることが望ましい。

【0136】 この観点から、縮退していない高さ関数をもつ $C^2$ 可微分の多様体をもとに機能の拡張を行う。具体的には以下のとおりである。

【0137】 ◎ 微分不可能な点を含む輪郭線

図18は微分不可能な点を含む物体を示す。この場合、図19に示すような微分可能な形状関数をもつ物体に置き換えることによって符号化が可能となる。

【0138】 ◎ 水平な頂部、底部または分岐部

水平面を表す場合には、高さ $h$ の変化を0に置き換えればよい。例えば図20に示すように、頂部も分岐部もそれぞれ水平面の場合、図21のように高さを0として表現することができる。

【0139】 ◎ 尾根線 (リッジ)

図22に示すようにリッジに対応する場合、形状関数 $f$ および $g$ を、途中で折り返す線分 $\alpha : I \rightarrow R^3$ を用いて、以下のように設定することができる。すなわち、1本のひもを2つ折り畳んだような形状である。

$$【0140】 f(x), g(x) = \alpha(2x) \quad 0 \leq x \leq 1/2$$

$$f(x), g(x) = \alpha(1-2x) \quad 1/2 < x \leq 1$$

◎ 火山のリム

図23に示すように火山のリムについては、リッジ同様の方法および高さ $h$ を0にすることによって記述が可能となる。

【0141】 [3] 断面データからの滑らかな曲面の生成

(1) はじめに

断面データからもとのオブジェクトを再構成したい場合がある。特に医療分野では、いくつかの断面データからもとの器官の全体形状を再生する意義が大きい。3D医療データの生成方法として次のサーフェスモデルが知られている。

【0142】 1. 三角形タイル手法

10 多角形で表現された隣接輪郭線どうしの間を三角形パッチで埋めていく。

2. スプライン近似

スプライン近似によって曲面を構成する。

ここではホモトピー、および特別な場合としてこれら2つのサーフェスモデルを包含する新たなモデルを提案する。このモデルは、これら2つのサーフェスモデルの欠点を解消するものである。

【0143】 このホモトピーモデルは、連続トロイダルグラフによる輪郭線上の対応点対の表示と、その表示からホモトピーを用いて曲面を生成するステップを含む。最終的に得られる曲面は古典的なパラメトリックな曲面であり、スプライン近似を包含している。ここでは2つの例を示す。1つは直線ホモトピーを用いてロフト曲面を洗練するもの、もう1つはカージナルスプライン曲面である。

【0144】 (2) トロイダルグラフによる表現

30 三角形手法については、トロイダルグラフによる表現が知られている。このグラフ理論では最短対角線アルゴリズムに基づく手法が提案されている。まず始めに、既知のトロイダルグラフ (これは連続グラフではない) について説明し、つぎに我々が拡張した連続バージョンを説明する。

【0145】 ここで、輪郭線はひとつづきの線分によって近似されるものとする。また、ある輪郭線を $m$ 個の異なる輪郭線点 $P_0, P_1, \dots, P_{m-1}$ の列によって定義し、別の輪郭線を $Q_0, Q_1, \dots, Q_{n-1}$ によって定義する。これら2つの閉曲線の方法は同じであると考えられる。三角形の生成は次の2つの条件を満たさなければならない。

40 【0146】 1. 同一輪郭線上の2つのノードが同じ三角形のノードとして定義されるためには、これらのノードがその輪郭線上で隣合っていなければならない。

【0147】 2. どの三角形についても、頂点のうちの2つだけが同じ輪郭線から取得される。

【0148】 従来のトロイダルグラフでは、これら2つの規則を統合した上でグラフ理論に反映されている。すなわち、三角形相互の位相上の関係をトロイダルグラフの中に表現するのである。このグラフでは、頂点は $P_0, P_1, \dots, P_{m-1}$ と $Q_0, Q_1, \dots, Q_{n-1}$ の間の可能なスパンすべての組に対応し、弧は可能な三角形すべての組に対応する。例えば、頂点 $(P_3, Q_5)$ は $P_3$ と $Q_5$ を結

ぶスパンを示し、頂点  $(P_3, Q_5)$  から  $(P_3, Q_6)$  の弧によって  $\triangle P_3 Q_5 Q_6$  の三角形ができる。

【0149】曲面として許容可能なもののグラフでは、すべての行において垂直方向の弧がちょうど1つ存在し、またすべての列において水平方向の弧がちょうど1つ存在する。輪郭線間に生成される曲面のうち、許容できるものは2とおりである。1つは円筒と位相同形のもの、もう1つは2つの円錐と位相同形のものである。以下、前者の場合を中心に説明する。

【0150】最短対角線アルゴリズムは単純では、まず隣接する輪郭線がそれぞれ単位正方形に写像される。ここで点  $P_0$  と、 $Q_0$  が近いと仮定する。最短対角線アルゴリズムでは、点  $P_0$  と点  $Q_0$  を結ぶことによって三角形の生成を開始する。点  $P_1$  および  $Q_1$  が接続された後、つぎに接続すべき候補である  $P_1 - Q_{j+1}$ 、または  $P_{i+1} - Q_j$  の一方が選択され、接続される。以降、この繰り返しである。これを平面上で考えれば、階段関数によって表現できる。ただし、輪郭線を一周すればもとに戻るため、階段の開始点と終了点は同一点でなければならない。このことを表現するために、 $P$  をトーラスの外周を円周に沿って移動する方向、 $Q$  をそれと垂直方向にとることができる。この表現方法の場合、 $P$  がトーラスの外周を一周したとき、 $Q$  もトーラスの曲がった円筒部分を一周し、もとの点に戻る。

【0151】以上、この方法は、隣接する閉曲線どうしの形状が近い場合には極めて良好な結果をもたらすが、それらの形状が大きく変化する場合には、無理な対応づ

$$\begin{aligned} \cdot \text{定義1: } d(P_i, Q_j) &= \min d(P_i, Q_k) \quad (0 \leq k < n) \\ d(P_i, Q_j) &= \min d(P_k, Q_j) \quad (0 \leq k < m) \end{aligned}$$

がともに成り立つとき、頂点  $(P_i, Q_j)$  を最近点対とよぶ。ここで、 $d(P, Q)$  は点  $P$  と点  $Q$  の間の距離である。

【0156】・注意1：トロイダルグラフの各行および各列に存在する最近点対は、せいぜい1つである。

【0157】一般に、すべての最近点対を結ぶ許容可能な経路は存在しない。そのためできる限り多くの最近点対を結ぶ方法が検討される。そのためまず、最近点対である点の重みを1、それ以外を0とする。 $\Psi(S)$  を  $S$  が通過する頂点の重みの合計と定義する。いいかえれば、 $\Psi(S)$  は許容可能な経路  $S$  が通過する最近点対の数である。この結果、問題はコスト許容性が最大になる経路を見つけることに還元される。

【0158】もちろん理論的にはそれで正しいが、そのための計算量は非常に大きい。そこで我々は、最近点対をグループ化する、より単純なアプローチを考案した。実験の結果、最近点対がグループ化できることが判った。そのグループとは、 $P_i$  と  $Q_j$  以降、それぞれ次の点  $(P_1, Q_j)$ 、 $(P_{i+1}, Q_{j+1})$ 、 $\dots$   $(P_{i+k}, Q_{j+k})$  から構成されるようなグループである。これを「長さ  $L(R)$  が  $k+1$  である最

けが原因となり、生成される曲面に人工的な皺、またはフォールド(折り目)が生じることが知られている。

【0152】この様子は図24の  $q_3 - p_0 - q_2$  に示される。同図のごとく、曲面は点  $A$  の回りにしわを持つように見える。人工的な皺はスムーズシェーディングを用いても消すことができない。この問題はノードの接続状態を修正することで解決することはできない。なぜなら、 $p_0$  および  $p_1$  の間にノードが存在しないためである。

【0153】このように、三角形手法によって生成される曲面は各輪郭線に対するノードの選択に大きく依存する。三角形手法は、輪郭線を局所的に見ながら三角形をつないでいくため、上記の皺のような局所の問題が生じる。なお、各輪郭線の外周がちょうど1になるように輪郭線を正規化することによって、輪郭線間のコヒーレンス(接続性)を改善する提案もある。しかしその方法でも輪郭線上の局所的な制約条件のみが考慮されている。三角形手法では、生成される曲面の滑らかさも問題になる。スムーズシェーディングを施された三角形表現の曲面は一見滑らかに見えるが、曲面の法線は曖昧になり、オブジェクトの形状が複雑な場合、それを正確に把握することが困難になる。

【0154】(3) 最適経路を見出すための発見的アプローチ

以下、経路とは対応しあう輪郭線上の対応しあう点を結ぶ線をいう。ここではまず、「最近点対」を定義する。

【0155】

近点対のひとづづきのラン  $R$  と表現し、点  $p$  がラン  $R$  の構成要素であるとき  $p \in R$  と表記する。このとき次の命題が成り立つ。

【0159】命題1. 最近点対  $(P_i, Q_j)$  を通過する許容可能な経路  $S_1$  が存在するとき、ラン  $R$  のすべての最近点対を通過する許容可能な別の経路  $S_2$  が存在する。ただし、 $(P_i, Q_j) \in R$  であり、 $\Psi(S_1) \leq \Psi(S_2)$  である。

【0160】[証明]  $L(R) \geq 2$  であるとして、一般性を失わない。

【0161】ケース1:  $(P_{i+1}, Q_{j+1}) \in R$ 、かつ  $S_1$  がこの点を通らない場合

ケース1a:  $S_1$  が  $(P_{i+1}, Q_j)$  を通る場合

$S_1$  が列  $j+1$  を離れるとき、最後に通る点を  $(P_k, Q_{j+1})$  とする。また、 $S_1 = S_3 S_4 S_5$  とする。ここで  $S_4$  を  $(P_{i+1}, Q_j) \sim (P_k, Q_{j+1})$  の経路とすれば、 $S_4$  は  $(P_i, Q_j)$  も  $(P_{i+1}, Q_{j+1})$  も通らないため、注意1により  $\Psi(S_4) = 0$  となる。一方、 $(P_{i+1}, Q_{j+1}) \in R$  であるため、明らかに  $(P_{i+1}, Q_{j+1}) \sim (P_k, Q_{j+1})$  の経路であって  $\Psi(S_6) = 1$  となる  $S_6$  が存在する。それゆえ、 $S_2$  を  $S_3, S_6, S_5$  を含む(す

なわち、 $S_4$ のかわりに $S_6$ を含む経路とすれば、 $\Psi(S_2) = \Psi(S_1) + 1$ である。

【0162】ケース1b:  $S_1$ が $(P_i, Q_{j+1})$ を通る場合

ケース1aと同じである。

【0163】ケース2:  $(P_{i-1}, Q_{j-1}) \in R$ 、かつ $S_1$ がこの点を通らない場合

ケース1と同様である。

【0164】この構成方法を再帰的に適用することにより、命題1の経路 $S_2$ が得られる。命題1によれば、 $\Psi$ を最大にするためにランのみを考慮すればよく、各ランの中身までは考えなくてよいことが判る。したがって前に述べた目的、すなわち、できる限り多くの最近点対を結ぶ方法の検討は、 $\Psi$ を最大にする許容可能なランの集合を選択することに還元される。

【0165】ここで、許容可能な経路が通過するランの集合を許容可能なランの集合と定義する。明らかに、2つのランから構成されるランの集合はいずれも許容可能である。3つのラン $\{R_1, R_2, R_3\}$ の集合であって許容可能でないものは以下のとおりである。まず、 $R_1 = (P_{i1}, Q_{j1}) \dots$ 、 $R_2 = (P_{i2}, Q_{j2}) \dots$ 、 $R_3 = (P_{i3}, Q_{j3}) \dots$ 、および $i_1 < i_2 < i_3 \pmod{m}$ と仮定する。ここで、 $j_1 < j_3 < j_2 \pmod{m}$ または $j_2 < j_1 < j_3 \pmod{m}$ または $j_3 < j_2 < j_1 \pmod{m}$ であるとき、この集合は許容可能ではない。 $m$ の剰余系を考えるのは、輪郭線が一周するとともに戻るためである。

【0166】4以上のランで構成される集合については、それに含まれるいかなる3つの構成要素も許容可能な集合である場合に限り、許容可能である。このことからわかるように、この計算労力は大きい。

【0167】こうした複雑性に鑑み、我々は発見的なスキームを考えることにした。複雑さの原因の1つは $m$ に関する剰余を用いる点にあるため、これを廃止する。これはトーラス上に $P, Q$ を表したとき、そのトーラスを切断して平面にすることに対応する。この単純化により、許容可能性はランの三値的な問題ではなく、二値的な問題になる。2つのラン $(P_{i1}, Q_{j1}) \dots$ 、および $(P_{i2}, Q_{j2}) \dots$ の集合は、 $i_1 < i_2$ かつ $j_1 < j_2$ 、または $i_1 > j_2$ かつ $j_1 > i_2$ の場合、許容可能である。集合 $\{R_1, R_2\}$ が許容可能でないとき、 $R_1$ は $R_2$ と不適合であると表現する。このテスト方法は極めて簡単であるが、トーラスを2つに切断して平面にすると、ランが分割される可能性がある。そこで最終的に、 $\Psi$ を最大にするランを選択するために、スタックを用いた以下の発見的アルゴリズムを採用する。

#### 【0168】アルゴリズム

$N$ 個のランがあり、各ラン $R_k = (P_{ik}, Q_{jk})$ について $i_1 < i_2 < \dots < i_k < \dots < i_n$ が成り立つとする。また最初にスタックは空であるとする。

【0169】ステップ1.  $R_1$ をスタックに積む。

ステップ2.  $k$ を2とする。

ステップ3.  $R$ をスタックの一番上に積まれたランであるとする。

ステップ4. もし $R_k$ と $R$ が適合すれば、 $R_k$ をスタックに積む。

そうでない場合であって、かつ $L(R_k) \leq L(R)$ であれば、 $R_k$ がスタックから取り除かれ、 $k = k + 1$ とする。それ以外の場合、 $R$ がスタックから取り除かれる。

【0170】ステップ5.  $k \leq N$ であればステップ3に戻る。

【0171】 $R_k$ については、 $k$ 回以下の繰り返しが必要である。そのため、このアルゴリズムの時間複雑性は $O(N^2)$ である。このアルゴリズムにより、互いに適合するランのうち、ある長さを超えるものが次々にスタックに積まれ、確保されていく。この結果、 $\Psi$ をほぼ最大値にする許容可能なランの集合を得ることができる。最終的に得られたランの様子は図25に示される。

#### 【0172】◎ トロイダルグラフの連続バージョン

つぎに、経路の進路を詳細に決める必要がある。経路は上述のアルゴリズムによって選ばれたランの各点を通過しなければならないが、それ以外の点についてはまだ未定の状態にある。ここでは進路を決めるために、最近点対どうしの間の線形補間を採用する。詳細な説明に入る前に、離散的なトロイダルグラフの連続バージョンの概要を説明する。

【0173】まず始めに、上下の輪郭線はなんらかのパラメータによって表示されなければならない。輪郭線上の点は以下の関数によって指定される。

【0174】 $f, g: I \rightarrow R^3$ 、ここで $I \in R$ 、かつ $[0, 1]$ で定義される。また $f(0) = f(1)$ 、および $g(0) = g(1)$ である。

【0175】連続トロイダルグラフでは、そのグラフに含まれる2つの点の水平および垂直距離がそれら2つの間のパラメータ値の差を表す。したがって、パラメータがNURBS曲線などのパラメータそのものを指す

【2】とは意味合いが異なる。パラメータの例は以下のとおりである。

【0176】・例1: 弧長をパラメータとして使う。 $l_1$ と $l_2$ がそれぞれ下と上の閉曲線の長さを表すとする。

例えば、 $(P_{i1}, Q_{j1})$ と $(P_{i2}, Q_{j2})$ のグラフ上の水平距離は上の閉ループ上の $P_{i1} \sim P_{i2}$ の弧長を表す。またグラフ上の垂直距離は下の閉ループ上の $Q_{j1} \sim Q_{j2}$ の弧長を表す。グラフ上の点 $(x, y)$ は $P_0$ からの弧上の距離が $x \cdot l_1$ であるような点 $P$ 、および $Q_0$ からの弧上の距離が $y \cdot l_2$ であるような点 $Q$ の組を表すよう正規化される。

【0177】・例2: 輪郭線の形状が円に近い場合、偏角をパラメータに用いることができる。この場合、本質的には円筒座標系を用いることと同じである。なお、輪



郭線は線分で近似する必要はない。パラメトリックな曲線、例えばベジェ曲線、Bスプライン曲線またはカージナル・スプライン曲線を利用することができる。

【0178】・例3：スプライン基数関数のパラメータをグラフのパラメータとして利用する。

【0179】後述する線形補間を利用するためには、弧長による表示が望ましい。しかし、曲線パラメータから弧長への変換は容易ではない。1つの解決策は、スプライン曲線を線分によって近似し、実際の弧長の代りに線分の長さを利用する方法である。後に示す表示例はこの近似を用いている。

【0180】点が輪郭線上を最初のノードから離れるに従ってパラメータが単調増加する限り、以下の議論においてどのようなパラメータを使うかは本質的な問題ではない。許容可能な経路に関するグラフの連続バージョンは、このグラフ上で単調増加（または単調減少）する「多値関数」として表現される。厳密に言えば、経路はグラフ  $y = U_i(x)$  および  $x = V_i(y)$  の連鎖として表示される。ここで、

$$U_i : [x_{2i}, x_{2i+1}] \rightarrow I$$

$$V_i : [y_{2i}, y_{2i+1}] \rightarrow I$$

( $0 = x_0 \leq x_1 \leq \dots \leq x_{2n-1} < 1$ ,  $0 = y_0 \leq y_1 \leq \dots \leq y_{2n-1} < 1$ ) は、単調増加（単調減少）関数であり、 $U_i$  および  $V_i$  の結合点  $(x_{2i+1}, y_{2i+1})$  において、

$$U_i(x_{2i+1}) = V_i(y_{2i+1}), \text{ および}$$

$$U_0(0) = V_{n-1}(1)$$

が成立し、 $V_i$  および  $U_{i+1}$  の結合点  $(x_{2i+2}, y_{2i+2})$  において、

$$V_i(y_{2i+2}) = U_{i+1}(x_{2i+2})$$

が成立するような関数である。

【0181】図26は  $n=2$  の場合を示す図である。なお、離散的なトロイダルグラフはこの連続バージョンの特別な場合であり、変換は階段状、すなわち  $y = q_i$  および  $x = p_i$  の連鎖という形で表される。

【0182】以上の考察をもとに、経路の詳細が最終的にこのグラフ上で決定される。最近点対が  $(x_i, y_i)$  ( $i = 0, 1, \dots, k-1$ ) であり、 $0 = x_0 < \dots < x_{k-1}$  であると仮定する。この経路は以下のような関数  $U(x) : I \rightarrow I$  によって記述することができる。

【0183】

【数12】

$$U(x) = (y_{i+1} - y_i) \frac{(x - x_i)}{(x_{i+1} - x_i)} + y_i (x_i \leq x < x_{i+1}).$$

すでに述べたとおり、この式は  $(x_i, y_i)$  と  $(x_{i+1}, y_{i+1})$  の間の線形補間である。この様子は図27に示される。この経路によって表現される表面は図28に示される。輪郭線自身は図24と同じである。この例は直線ホモトピーを表している。図24と図28を比較すれば明らかとなお、図28の表面の方が図24

に比べてより滑かである。「皺」の発生も回避されている。

【0184】なお、定義1の最近点対は離散的な点どうしの距離の比較から決定したが、この概念も連続トロイダルグラフに合わせて拡張することができる。すなわち、最近点対を「輪郭線上で互いに最も近い点の対」と定義すればよい。ただし、最近点対は無数に存在しうるため、実際のインプリメンテーションでは適度なサンプリングを行うものとする。

【0185】以上、我々の手法では、まず最近点対間の対応をとり、つづいてそれ以外の点の対応をとるというステップを経る。このため、局所的な対応状態に注目する従来の手法で問題となる不自然な接続結果が生じることはい。また、最初に輪郭線全周に渡って最近点間の対応をとるため、輪郭線の局所構造に起因するノイズに対する耐性が向上する。

【0186】(4) ホモトピーモデルに基づく表面の生成

すでに述べたとおり、連続トロイダルグラフ上の点は上下の輪郭線の点の間の対応を示す。対応点どうしはホモトピーによってつながれる。正確に言えば、曲面は  $f \sim g$   $U$  へのホモトピーの軌跡によって生成される。ここで  $U$  は連続トロイダルグラフ上の関数であり、輪郭線どうしの対応を示す。直線ホモトピーの場合、それらの点は単純に直線によって接続される。以下、ホモトピーモデルを用いた表面生成を説明する。

【0187】このホモトピーモデルでは、ホモトピーを用いて各輪郭線上の対応点どうしを接続することにより、隣接する輪郭線の間に表面パッチを生成する。このモデルでは離散的なトロイダルグラフによる対応点対の表現とは異なり、輪郭線上のすべての点が隣接する輪郭線に対応点を持つため、ホモトピーがそれらの対応点間を接続するファイバーとして利用される。輪郭線間の表面パッチはホモトピーの軌跡と見なすことができる。わかりやすい例は円筒である。円筒の表面は2つの円の間の対応点を接続することで生成できる。詳細は省くが、このモデルはパラメトリックな曲面を包含している。

【0188】◎ ホモトピーを用いたパラメトリックな曲面の生成

下および上の輪郭線を写像  $f, g : X \rightarrow R^3$ ,  $X = [x_0, x_1] \in R$  で表す。

【0189】簡単のために、下の輪郭線は  $z=0$  の平面上、上の輪郭線は  $z=1$  の平面上にあるとし、 $X=I$  とする。「2つの点  $P(p_x, p_y)$ 、 $Q(q_x, q_y)$  がホモトピー  $F$  によってつながれる」とは、

$$F(x, 0) = f(x) = (p_x, p_y, 0)$$

$$F(x, 1) = g(x) = (q_x, q_y, 1)$$

がともに成立するような  $x \in I$  が存在することである。すなわち、上下の輪郭線の間にあり、平面  $z=t$  上のす

すべての断面が  $x \in X$  なる  $F(x, t)$  によって与えられるとき、 $F$  は  $f$  から  $g$  へのホモトピーである。いいかえれば、表面は2変数関数  $F(x, t)$  によって表される。

【0190】2つの輪郭線がともに1個の閉曲線からなる場合、それらは位相同形である。一方、2つの輪郭線が異なる数の閉曲線を含む場合、それらは位相同形ではない。そのような場合、分岐に対する取扱いが必要になる。簡単のために、上の輪郭線は2つの閉曲線、下の輪郭線は1つの閉曲線を含むと仮定する。分岐に対する取扱いは、2つの閉曲線が1点で結合するような仮想輪郭線を上下輪郭線の間に導入することによって可能となる。図29に示すとおり、この結合箇所が1点である場合、仮想輪郭線は2つの閉曲線を含む。またこの結合箇所が2つの別個の点と考えられる場合、仮想輪郭線は1つの閉曲線からなると考えてよい。

【0191】◎ トロイダルグラフ表示からの表面の生成

$f, g: I \rightarrow R^3$  が下および上の輪郭線を表す関数とする。ホモトピーモデルの場合、考慮すべき許容可能な経路は逆関数  $U^{-1}$  を持つ単調増加連続関数  $U: I \rightarrow I$  によ

ってトロイダルグラフ上に表現されなければならない。

$$(w_{-1}(t)w_0(t)w_1(t)w_2(t)) = (t^3t^2t^1) \begin{pmatrix} -a & 2-a & -2+a & a \\ 2a & -3+a & 3-2a & -a \\ -a & 0 & a & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

である。 $f_1$  がカージナルスプライン関数によって表現されるとき、その表面をカージナルスプライン表面と呼ぶ。

【0194】[4] 断面からのレーブグラフの自動的な生成

(1) はじめに

この章では、オブジェクトの断面データが与えられたときに、そのオブジェクトの符号を自動的に取得する方法を説明する。この技術により、断面データが利用できるとき、自然界に存在するオブジェクトの形状を取扱うことができる。まず最初に、オブジェクトのレーブグラフを自動的に再構成する。レーブグラフが構成されれば、そこから表面の符号を得ることは容易である。ここでも  $C^2$  連続性と非縮退の特異点を仮定する。

【0195】一連の処理は[2]で説明した符号化データから表面を生成する処理の逆演算と見なすことができる。このためユーザーは、表面を直接符号化する代りにレーブグラフまたは断面を記述することができる。この方法はまた、得られた結果が、直接断面データを用いた表面再構成システムに与えられる場合にも有益である。

【0196】医療分野への応用を考えたとき、表面再構成システムに対してデータを準備する工程は以下のとおりである。すなわち、CTスキャンで器官の画像を撮影して登録する。つぎに輪郭線の外形線をデジタイズし、いずれの輪郭線どうしを接続するかを決定する。この

しかる後、 $f(x)$  および  $g(U(x))$  の間のホモトピーによって表面が生成される。直線ホモトピーが使用されるとき、これは次の式で表現される。

【0192】

【数13】

$$\bar{F}(x, t) = \begin{cases} (1-t)f(x) + tg(0) & x \leq t \\ (1-t)f(0) + tg(x) & x > t \end{cases}$$

カージナルスプラインを用いることもできる。4つの輪郭線を  $f_{-1}, f_0, f_1, f_2$  で表し、それぞれの間の許容可能な経路を  $U_{-1}, U_0, U_1$  でそれぞれ表すとする。この場合  $f_0$  と  $f_1$  の間の表面は次の式で表現される。

【0193】

【数14】

$$F(x, t) = w_{-1}(t)f_{-1}(U_{-1}^{-1}(x)) + w_0(t)f_0(x) + w_1(t)f_1(U_0(x)) + w_2(t)f_2(U_1(U_0(x)))$$

ただし、

【数15】

$$(w_{-1}(t)w_0(t)w_1(t)w_2(t)) = (t^3t^2t^1) \begin{pmatrix} -a & 2-a & -2+a & a \\ 2a & -3+a & 3-2a & -a \\ -a & 0 & a & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix}$$

後、スムージングによって登録されていないデータを計算し、最後に表面を構成する。

【0197】この一連のプロセスの中で最も難しい部分は、いずれの輪郭線対に表面パッチを生成すべきかを決定することである。この目的のためにレーブグラフが利用される。レーブグラフでは各輪郭線がノードとして表現され、2つの連続する断面の輪郭線間の位相上の関係を示すことができる。

【0198】既述の図4(a)～(c)は、例としてトラスの高さ関数のレーブグラフを示す。この例では、レーブグラフの点は断面上の輪郭線を表している。例えば人の内耳の半規管のように、データが複雑な場合、このレーブグラフを人手作業で生成することは大変な労力を要するとともに、非常に難しい作業になる。圧倒的な枚数の断面画像からオブジェクトの全体的な位相構造を理解することはほとんど不可能に近い。

【0199】位相構造の理解を容易ならしめるために、我々はレーブグラフの自動再構成方法を開発した。隣接する断面上に存在する任意の輪郭線対を考えると、この対の重み関数を定義する。我々の方法は、この重み関数および表面にあいたハンドル(つまり穴)の数に関する先験的な知識を用いることによって、レーブグラフを自動的に構築することができる。まず始めに、我々のアルゴリズムは輪郭線の数が増えないような、レーブグラフの辺の主要部分を生成する。その後、上述の重み関

数およびハンドルに関する先験的な知識を用いてグラフの残りの部分を決定する。具体的には、既知のハンドルの数に矛盾しないような辺を、重みが減っていく順に（すなわち、最初に重みが最大の箇所に辺を付けるように）、グラフに付け加えることによってレーブグラフを完成させる。

#### 【0200】(2) レーブグラフの定義

レーブグラフは立体オブジェクトの位相上の「骨格」を表し、いずれの輪郭線の間に表面パッチを生成すべきかを示す。

【0201】レーブグラフは多様体の上に定義される。多様体は、ユークリット空間と同じ局所的特性を持つとみなすことができる。例えば、平面や球面は二次元多様体である。以下、立体オブジェクトの表面は二次元多様体であると仮定する。

【0202】ここでは、オブジェクトの二次元多様体の表面上の高さ関数  $h(X)$  に関するレーブグラフを用いる。ここで  $h(X)$  はオブジェクトの表面上の点の高さを与える関数で、 $X = (x, y, z)$  は  $R^3$  ( $x, y, z \in R$ ) に埋め込まれている。すなわち、 $h(x, y, z) = z$  である。

【0203】表面上の高さ関数のレーブグラフは、 $R^3$  におけるグラフの商空間である。商空間は2つの点  $(x_1, y_1, z)$  および  $(x_2, y_2, z)$  が高さ  $z$  において曲面の断面上の同じ連結成分であるとき、これらを等化する（図4(a)～(c)）。これは、図4(b)に示すように  $z$  軸に対して垂直な断面を考えればわかりやすい。すなわちレーブグラフは、各断面における輪郭線をノードとして表現する。オブジェクトの表面に開いている穴（ハンドル）はすべての臨界点为非縮退であり、かつ各鞍点に集まる辺の数が3であるとき、レーブグラフにおいて閉曲線として現れる。この知見を後述のアルゴリズムに利用する。

【0204】レーブグラフは、いずれの輪郭線の間に表面パッチが存在するかを示している。以下、底部から  $i$  番目の断面を含む平面の式を  $z = z_i$  と仮定し、この平面を第  $i$  フレームと呼ぶことにする。

#### 【0205】(3) レーブグラフの構成

##### ◎ 重み関数

ここでは一連の断面が与えられたときにレーブグラフを構成する重み関数を定義する。隣接する断面に含まれる輪郭線の対が与えられたとき、この関数は重みを与える。分岐を取扱うとき、重みが減っていく順に輪郭線の間に辺を張っていく。

【0206】重み関数の作り方には何とおりかの方法がある。重み関数は、互いにより近い輪郭線の対に対してより大きな重みを与えるものとする。おおまかにいえば、我々は輪郭線間の「平均距離」の逆数を用いた。以下の計算において、我々はこの平均距離を輪郭線間の距離  $D$  によって割った。このことにより、異なる断面の輪

郭線の対の重みを比較することができる。

【0207】第  $j$  フレームの第  $i$  輪郭線を  $L^j_i$  で表現する。また、 $L^j_i$  および  $L^{j+1}_k$  の間に生成される表面パッチに対応する辺の重さを  $q(L^j_i, L^{j+1}_k)$  で表す。 $N_j$  は第  $j$  フレームに含まれる輪郭線の数、 $N_{frame}$  はフレームの数を表す。輪郭線は多角形によって近似されるものとし、実際のインプリメンテーションでは以下の式1を用いた。

#### 【0208】

10 【数16】

$$q(L^j_i, L^{j+1}_k) = n^j_i D_j / \sum_{m=0}^{n^j_i} f(x^j_{i,m}, L^{j+1}_k).$$

ここで  $D_j$  は第  $j$  フレームと第  $(j+1)$  フレームの間の距離である。 $L^j_i$  を近似する多角形は  $n^j_i$  個の頂点を持ち、第  $m$  点を  $x^j_{i,m}$  で表す。 $f$  は次式で定まる。

#### 【0209】

【数17】

$$f(x, A) = \min_{y \in A} d(x, y)$$

この式において  $d$  は距離の関数、 $A$  は  $R^3$  に含まれる点の集合、 $x, y \in R^3$  とする。この関数  $f$  は  $x$  と集合  $A$  の中の点の最短距離を与える。

#### 【0210】◎ レーブグラフの自動生成

以下、重み関数と任意の一連の断面を用いて我々のアルゴリズムによるレーブグラフの構成を説明する。始めに、つぎに示すケース1のごとく、輪郭線の数が変わらない主要な部分についてグラフを生成する。数が変わる場合は、ケース2で対応する。すなわち、予めわかっている穴の数と矛盾しない辺を、重みの減る順番にレーブグラフの中に張っていく。

#### 【0211】ケース1: $N_j = N^{j+1}_j$ の場合

ケース1a: 輪郭線が互いに最も近い  $L^j_{i1}, L^{j+1}_{i2}$  の対で構成される場合。

すなわち、

#### 【数18】

$$q(L^j_{i1}, L^{j+1}_{i2}) = \max_{k=0}^{N^{j+1}_j-1} q(L^j_{i1}, L^{j+1}_k)$$

$$q(L^{j+1}_{i2}, L^j_{i1}) = \max_{k=0}^{N^j_j-1} q(L^{j+1}_{i2}, L^j_k)$$

がともに成り立つ場合（このような対を「最近輪郭線対」と呼ぶ）。

【0212】重み関数を用いることにより、一対の輪郭線の間が辺で結ばれる。図30(a)～(e)はケース1、2を説明する図である。ここでは、図30(a)および(b)のような状況は発生しないと仮定する。

【0213】ケース1b: ケース1aが成り立たない場合

図30(c)に示すような分岐が生じていると考える。

我々のインプリメントテーションでは、次の条件のいずれかが成立する場合、 $L^{j+1}_{i1}$  および  $L^{j+1}_{i2}$  の間に辺が張られる。

【0214】

【数19】

$$q(L^{j+1}_{i1}, L^{j+1}_{i2}) = \max_{k=0}^{N^{j+1}-1} q(L^{j+1}_{i1}, L^{j+1}_k)$$

$$q(L^{j+1}_{i2}, L^{j+1}_{i1}) = \max_{k=0}^{N^{j+1}-1} q(L^{j+1}_{i2}, L^{j+1}_k)$$

ケース2:  $N^j \neq N^{j+1}$  の場合

ケース2a: 2つのフレームの間に分岐が起っている場合 (図30(d))

ケース2b: 分岐が起っていない場合 (図30(e))

これら2つのケースは重み関数によって区別することができる。 $q(L^{j+1}_{i1}, L^{j+1}_k)$ 、 $q(L^{j+1}_{i2}, L^{j+1}_k)$  がともに大きな数値の場合、 $L^{j+1}_k$  から  $L^{j+1}_{i1}$ 、 $L^{j+1}_{i2}$  への分岐が起っている可能性が高い。

【0215】このように重み関数のみからでも分岐の可能性を判断することは可能である。しかし我々は、重み以外にオブジェクトの大域的な情報を利用するアルゴリズムを提案する。以下、 $N_{\text{conn}}$  はオブジェクトの連結成分の数、 $g$  はオブジェクトの穴 (ハンドル) の数、 $M_{\text{conn}}$  はレーブグラフの連結成分の数、 $l$  はレーブグラフの閉曲線の数を示す。つぎに示す9ステップのアルゴリズムでは、 $N_{\text{conn}} = 1$  であると仮定する。また最初、グラフには辺がない状態であるとする。

#### 【0216】アルゴリズム

・ステップ1: 隣接するフレーム上の輪郭線のすべての対について関数  $q$  を計算し、すべての輪郭線  $L^f_c$  ( $f$  はフレーム番号、 $c$  は輪郭線番号) についてその最も近くに隣接する輪郭線  $L^{f'}_{c'}$  ( $f'$  はフレーム番号+ $d$ 、 $c'$  は最近輪郭線番号) を示す表を作成する。 $L^f_c$  は  $q$  の最大値および  $d = \pm 1$  における  $q$  の値を与える。

【0217】・ステップ2: まずケース1のすべての対に関して辺をグラフに追加する。

【0218】・ステップ3: つぎにケース2にあたる輪郭線  $L^{j+1}_{i1}$  および  $L^{j+1}_{i2}$  の対を選択し、最近輪郭線対の間に辺を張る。

【0219】・ステップ4:  $1 > g$  または  $M_{\text{conn}} < N_{\text{conn}}$  の場合、アルゴリズムはステップ9に飛ぶ。

【0220】・ステップ5: ステップ3において辺が張られなかったケース2の輪郭線の対のリストを重みが大きい順にならべて生成する。こうして表示される辺を  $E_1$ 、 $E_2$ ... と書く。

【0221】・ステップ6: 作成されたリストの中から一番上にある輪郭線の対を選択する。

【0222】・ステップ7: その対に対して辺を張ったとき、 $1 \leq g$  かつ  $M_{\text{conn}} \geq N_{\text{conn}}$  なら、その辺をグラフに追加する。そうでない場合、その辺を拒絶する。

【0223】・ステップ8: 処理が終了した対をリストから削除する。リストが空にならなければ、ステップ6に戻る。

【0224】・ステップ9: 生成されたグラフが期待どおりでない場合、接続されるべきでない輪郭線の対の重みを対話形式で0に修正する。しかる後、処理をステップ1に戻す。

【0225】以上のアルゴリズムでレーブグラフが構成できれば対応する輪郭線が判明するため、後は【3】の表面構成技術によってオブジェクトを再構成することができる。なお、このアルゴリズムを  $N_{\text{conn}} > 1$  である複数要素のオブジェクトに適用する場合、ステップ4において各連結成分に対して  $g$  を指定すればよい。

#### 【0226】◎ 医療分野への適用

我々はこのアルゴリズムを用いて、人の耳の蝸牛管および半規管の三次元表面を再構成した。セロイジン (切片固定剤) 処理により、耳の標本を固定した上で、これを  $20 \mu\text{mm}$  の厚さにスライスした。この標本を写真撮影し、グラフィカ社のドラムスキャナ (G-225C) を用いて写真をイメージデータに変換し、スタイラスペンを使ってオブジェクトの外形線をプロットした。処理には、HP9000シリーズのモデル550を用いた。最後に輪郭線データをシリコングラフィックス社のパーソナルアイリスに入力し、レーブグラフを生成してオブジェクトの表面を再構成した。

【0227】図31は断面データを基に構成された蝸牛管のレーブグラフを示す。同図のごとく、渦巻状の構造が正しく表現されている。この例では穴の数  $g = 0$  である。

【0228】この例の場合、まずはじめに図31において太線で描かれている辺が生成された。つづいて辺  $E_1$  および  $E_2$  が追加され、 $E_3$  は拒絶された。なぜなら  $E_3$  はグラフにおいて閉曲線を作るためである。同様の手順で  $E_4$  が追加され、 $E_5$ 、 $E_6$ 、 $E_7$ 、 $E_8$  が拒絶された。処理はここで終了し、グラフは我々が予め知っている形状に一致した。

【0229】図32は内耳の半規管に関するレーブグラフである。半規管は穴を3つ持っている。この場合、辺  $E_1$ 、 $E_2$ 、...  $E_8$  および  $E_9$  が追加され、 $E_{10}$ 、 $E_{11}$ 、 $E_{12}$  が拒絶された。その結果、グラフには正しく3つの閉曲線が生じた。

【0230】この自動化アルゴリズムが、通常人が目で行う方法よりも優れている理由は、十分な数の断面データがあるとき分岐が生じるような輪郭線に対して  $q$  の値が大きくなるため、分岐を正しく扱うことができる点にある。この情報は、単に断面図を見ているだけでは得ることができない。現実には我々が実験を行ったとき、外部からの指示は全く不要であった。

【0231】(4) レーブグラフを用いた表面の符号化一旦レーブグラフが構成できれば、表面の符号を得るこ

10

20

30

40

50

とは容易である。したがって、表面の符号化を直接行う代りに、レーブグラフまたは断面データを記述することによってオブジェクトを符号化することができる。符号はレーブグラフを上から下にスキャンすることにより、以下の手順で得ることができる。

【0232】1. ある高さにおいてグラフのノードが新たに現れたとき、対応する符号はPut\_e2である。このノードが既存の輪郭線（輪郭線#nとする）に含まれていれば、対応する符号はPut\_e2(n)である。

【0233】2. ある高さにおいてグラフのノードが消えたとき、対応する符号はPut\_e0である。

【0234】3. 2つの辺がマージされる時、対応する符号はPut\_el\_mergeである。新たな輪郭線の子輪郭線がマージされた2つの輪郭線の子輪郭線に一致しない場合、エラーが報告される。

【0235】4. 辺が分れるとき、対応する符号はPut\_el\_divideである。

【0236】分離した輪郭線の子輪郭線（第2パラメータclist）は、輪郭線どうしの包含関係をチェックすることで決まる。分岐した一方の輪郭線（#mとする）が他方（#nとする）に含まれる場合、[1]で説明した第3パラメータはoutsideとなる。それ以外（兄弟輪郭線）の場合、そのパラメータはinsideとなる。outsideの場合、#mの子輪郭線が#nの子輪郭線になるかどうかチェックされる。一方、insideの場合、#mおよび#nの子輪郭線が元の輪郭線の子輪郭線に一致するかどうかチェックされる。

【0237】ある高さにおいて符号が計算された後、その高さにおける輪郭線どうしの親子関係が、対応する演算子を輪郭線の木構造に対して適応した結果得られるものと一致するかどうかチェックされる。もし一致しなければ、エラーである。親子関係を破っている辺の1つの重さは0に修正され、自動化アルゴリズムが再実行される。

【0238】[実施形態] 以上が前提技術である。この前提技術を本発明との関連を中心にまとめれば以下のとおりである。

#### 【0239】[1]について

モース理論は特異点と指数の情報しか与えない。したがって、立体を構成するための胞体（セル）の種類は概念として判明するが、それを具体的にどのような幾何的關係をもって接続していくかは不明である。レーブグラフもモース理論の問題点を解決したわけではない。レーブグラフでは、特異点相互の關係はわかるが、結び目や階層關係は判明しない。結び目を知るためには、埋め込みに関する情報、すなわち幾何情報を知る必要がある。以上の問題点は、数学上の目的はともかく、立体を正しく符号化するという産業上の目的からすれば解決が必須である。

【0240】こうした観点から、輪郭線の木構造をもと

に輪郭線どうしの階層關係を表現することができ、かつ立体の埋込み狀態を視覚的に容易に指定することの可能なアイコン表示を導入した。このアイコン表示は立体を符号化するために必要かつ十分な4種類の演算子（Put\_e0 など）と関連づけられているため、ユーザインタフェースとしてはアイコンにより、またシステムの内部処理としては演算子により、立体の符号化を位相的な正しさを保証しながら行うことが可能となった。

【0241】この[1]の技術は、立体を新たに設計する場合を目的とする。したがって、既存の立体をその断面から自動的に再構成するものではない。一方、本発明はキープフレームを断面として立体を再構成することに等しい。したがって、[1]の技術は本発明の主たる手段にはならず、実施形態4においてひとつの応用例として利用される。

#### 【0242】[2]について

[1]によって立体が符号化された狀態を前提に立体の表面を生成する。[1]の符号化によれば、臨界断面はそれぞれ演算子（またはアイコン）に対応しているためすべて判明する。したがって、臨界断面の間をホモトピーによって接続すれば立体の表面を生成することができる。臨界断面の間で立体の位相が変化することはなく、一方、ホモトピーは立体の位相を変化させないためである。[2]の技術は[1]と関連しており、[1]同様に実施形態4で利用する。

#### 【0243】[3]について

断面データが与えられたとき、対応輪郭線対の間で対応しあう点を検出し、それらの輪郭線の間を表面を生成する。[3]では基本的に輪郭線の分岐は考慮せず、対応輪郭線対は判明しているものとして、これらの輪郭線における対応点対を探索する。

【0244】対応点対の検出にはトロイダルグラフを用いる。本発明では従来より提案されている離散タイプのトロイダルグラフを用いてもよいが、本発明者が提案した連続タイプのトロイダルグラフを用いることができる。より自然な対応点対を検出するためには後者が有利である。

【0245】連続トロイダルグラフを用いるとき、本発明者の提案する「最多かつ最長のランを通過する経路」を見出すアルゴリズムを用いてもよい。ランは計算量削減のために所定の最近点対をグループ化するもので、輪郭線対を結ぶ自然な経路を発見できるとともに、計算時間短縮の効果を併せもつ。

【0246】最近点対が判明すれば、それら対の間を補間することにより、一方の輪郭線 $f(x)$ 上のすべての点と他方の輪郭線 $g(y)$ 上のすべての点を対応づけることができる。 $x, y$ をそれぞれの輪郭線の弧上の距離とすれば、対応点対は $y=U(x)$ という関数 $U$ で表現できる。 $U$ が判明すれば、 $f(x)$ と $g(U(x))$ をホモトピー $F$ で結べば曲面が生成できる。

【0247】[3]の技術は[4]によって対応輪郭線対が判明した後、本発明の主たる手段として実施形態1、2で利用する。本発明ではアニメータの労力を軽減するためにキーフレームどうしの対応点の入力をアニメータに要求しないため、対応点対の自動検出に[3]を活用する。

#### 【0248】[4]について

[3]の準備として、隣接する断面間における対応輪郭線対を検出する。対応輪郭線対の検出とレーブグラフの生成は同義と考えられる。

【0249】対応輪郭線対の検出には場合分けがある。まず、隣接する断面のそれぞれに含まれる輪郭線の数に等しい場合、重み関数に従って近い輪郭線どうしが対応づけられる。輪郭線の数異なる場合には、オブジェクトの連結成分とハンドルの数を考慮しながら近い輪郭線どうしを関連づけていく。一連の処理は本発明者の提案するレーブグラフの自動生成アルゴリズムで実現できる。

【0250】ただし本発明では、ユーザによる指定を容易にするために、連結成分とハンドルの数に関する所定のデフォルト設定を行う。設定は、アニメーション製作を念頭においている。これらのパラメータのほか、

[2]で導入した輪郭線の階層構造を示す木構造をもとに自然消滅または自然発生すべき輪郭線の推定を行う。さらに、ユーザの意図を確認するための簡単な指示入力を受け付ける。

【0251】**実施形態1.** 前提技術では、立体の表面をホモトピーによって構成したとき、そうして得られる立体がそのまま目的のオブジェクトとなった。したがって、前提技術ではオブジェクトの三次元形状を静止画として表現することを主眼とした。

【0252】一方、本発明では、立体は最終的にアニメーションのフレームを生成するための手段として用いられるにすぎず、オブジェクトの二次元形状を動画として表現することに主眼がある。このために、前提技術における立体のz軸を空間軸から時間軸に変更する。実施形態1ではまず、本発明に係るアニメーションの自動中割システムを説明する。

【0253】本実施形態を実現するシステムのハードウェアは、自動中割を行うためのプラットフォームであるPCと、このPCに対してアニメータの描いたキーフレームの画像を入力するためのイメージスキャナだけでよい。キーフレームをPC上の描画ソフトウェアなどで作成する場合にはイメージスキャナも不要である。これ以外の構成としては、実際に作成したアニメーションの各フレーム画像を出力するプリンタと、アニメーションに関するデータを記録する磁気記憶装置などがあればよい。

【0254】図33は、本実施形態に係る中割システム2のソフトウェアモジュール構成図である。同図のごと

くこのシステムは主に、キーフレームの画像を受け付けるキーフレーム受付部4、複数のキーフレームをもとにアニメーション立体を生成するアニメーション立体生成部6、生成されたアニメーション立体を切断して中間フレームの画像を取得する中割部26からなる。

【0255】アニメーション立体生成部6は、隣接するキーフレーム間における対応輪郭線対を検出する対応輪郭線検出部8と、対応輪郭線対の間の表面をホモトピーによって生成する表面生成部20からなる。

【0256】対応輪郭線検出部8は、前提技術[4]を利用する。対応輪郭線検出部8は、連結成分に関する情報を処理する連結成分関連処理部10、輪郭線の木構造をもとに消滅する輪郭線の推定などを行う木構造関連処理部12、連結成分および木構造を考慮したうえで輪郭線間の対応の可能性をそれらの近さをもとに評価する輪郭線対応評価部14、評価の結果にもとづき対応する可能性が高い輪郭線対を表示するオプション機能である候補表示部16をもつ。

【0257】一方、表面生成部20は、対応輪郭線対の間の対応点対を前提技術[3]をもとに検出する対応点検出部22と、[3](4)「ホモトピーモデルに基づく表面の生成」を用いて拡張トロイダルグラフからアニメーション立体の表面を生成するホモトピー関連処理部24をもつ。以下、中割システム2によって実際に中割を行う手順を図34を用いて説明する。

#### (S2) キーフレームの入力

まず、ユーザは複数のキーフレームを入力する。キーフレームに描かれているオブジェクトの輪郭線は各種形状関数、例えば、点、円、多角形、ベジェ、NURBSなどによって近似する。関数はユーザが選択可能としてもよい。精度が要求される場合は、NURBS曲線など二次曲線を忠実に再現できる関数が選ばれる。

【0258】ここでは図35に示す2枚のキーフレームF0とF1が入力されたとする。これらのキーフレームには円形オブジェクト30(以下「Oc30」)と長方形オブジェクト32(以下「Or32」)が描かれている。キーフレームF0で画面左にあったOc30がキーフレームF1では右に移動し、形状も円から楕円に変わっている。キーフレームはフリーハンドで描画すればよい。

#### 【0259】(S4) パラメータの指定

つづいてユーザは各種パラメータを指定する。パラメータによる指定は以下の2つである。

#### 【0260】1. オブジェクトの連結成分の数 $N_{conn}$

#### 2. 輪郭線の木構造を考慮するか否か

連結成分とは、キーフレームに登場する物体の個数をいうが、ここでは2つのキーフレームにおける連結成分の大きいほうを $N_{conn}$ と決めることにする。なお、ユーザが連結成分の数を指定しなかった場合、システムは以下のデフォルト設定を行う。

【0261】  $N_{conn} = \max(N^{F0}, N^{F1})$

ただし、 $N^{F0}$ 、 $N^{F1}$  はそれぞれキープフレーム  $F0$ 、 $F1$  に含まれる輪郭線の数である。ここではユーザは指定をしなかったものとする。この場合、図35から  $N^{F0} = N^{F1} = 2$  であるから、 $N_{conn} = 2$  が設定される。この処理は図33の連結成分関連処理部12で行われる。

【0262】一方、木構造は前提技術[2]で親輪郭線、子輪郭線などのことばで定義した輪郭線どうしの包含関係である。木構造を考慮する場合、後の対応輪郭線対の検出の際、

・輪郭線が消えたとしたら木構造の末端にある子輪郭線(木構造のリーフ)である

・自然発生する輪郭線は既存の輪郭線の親輪郭線になれない

という規則が課される。なお、ユーザがこの指定をしなかった場合、システムはデフォルトとして、「木構造を考慮しない」を内部的に設定する。ここでもユーザは指定をしなかったものとし、木構造が考慮されないとする。この処理は図33の木構造関連処理部12で行われる。

【0263】(S6) 輪郭線間の対応の評価

つぎに、輪郭線対応評価部14において輪郭線間の対応の評価を行う。この評価の際、S4における指定の内容を考慮したうえで、キープフレーム  $F0$  と  $F1$  の間で対応しあう可能性のある輪郭線対をすべて抽出する。いま、連結成分の数  $N_{conn}$  は2であるため、可能性のある輪郭線対を( $F0$ に含まれる輪郭線、 $F1$ に含まれる輪郭線)と表記すれば、

組合せ1: ( $O_{c30}$ ,  $O_{r32}$ ) かつ ( $O_{r32}$ ,  $O_{c30}$ )

組合せ2: ( $O_{c30}$ ,  $O_{c30}$ ) かつ ( $O_{r32}$ ,  $O_{r32}$ )

の2組に限られる。仮に、

組合せ3: ( $O_{c30}$ ,  $O_{c30}$ ) かつ ( $O_{r32}$ ,  $O_{c30}$ )

を考えると、2つの輪郭線が同一の輪郭線に集まることになり、連結成分が1個になって  $N_{conn} = 2$  に反する。したがって、組合せ1、2のみとなる。なお、木構造については考慮しないため、これら2つの組合せがそのまま対応輪郭線対の候補となる。

【0264】つづいて、組合せ1、2のいずれが妥当であるかを判定する。この判定は、組合せ1、2の含まれる合計4つの輪郭線対のそれぞれについてその近さを評価し、最も近いもの、すなわち最近輪郭線対を求める。最近輪郭線対は前提技術[4]の重み関数を用いて見出すことができる。いまの場合、( $O_{r32}$ ,  $O_{r32}$ ) が最近輪郭線対になるため、これを含む組合せ2が自動的に対応輪郭線対の組合せとして選択される。

【0265】(S8) 対応輪郭線対の決定

S6で対応輪郭線対が一応見つけ出された。本システム

では、候補表示部16によってS6で決まった対応関係を図36のごとく画面に表示する。この対応関係が正しければユーザは「OK」を入力し、対応輪郭線対が決定される。ただし、S8はオプション機能として通常はスキップしてもよい。

【0266】(S10) 対応点対の検出

ここで処理は図33の表面生成部20に移る。まず、対応輪郭線対( $O_{c30}$ ,  $O_{c30}$ ) および ( $O_{r32}$ ,  $O_{r32}$ ) のそれぞれについて、輪郭線上の対応点対を検出する。

【0267】( $O_{c30}$ ,  $O_{c30}$ ) を例に説明する。図37に示すごとく、2つの輪郭線を同じ大きさの正方形に収まるように正規化する。輪郭線の形状はスプライン曲線やNURBS曲線で表され、上下の輪郭線の形状関数をそれぞれ  $f(x)$  と  $g(y)$  とおく。 $x$ 、 $y$  はそれぞれ弧上の距離である。この表記は前提技術[3]

(3)において、連続トロイダルグラフのパラメータに弧長をとる場合に対応する。

【0268】つづいて上下の輪郭線間の最近点対を求める。最近点対とは、対応輪郭線対A、Bそれぞれの上の点  $a$ 、 $b$  について、点  $a$  から見て輪郭線B上の最も近い点が点  $b$  であり、かつ点  $b$  から見て輪郭線A上の最も近い点が点  $a$  であるとき、これら点  $a$ 、 $b$  の対をいう。図38はこうして求められた最近点対  $x_i - y_i$  を示している。実際のインプリメントでは、上述のごとく、最長かつ最長のランを発見するアルゴリズムを活用すればよい。

【0269】この後、補間計算によって最近点対以外の点どうしの対応を決める。例えば図38の場合、上の輪郭線の  $x_0$  と  $x_1$  の弧上の中間点は下の輪郭線の  $y_0$  と  $y_1$  の弧上の中間点に対応するなどと考えればよい。輪郭線上のすべての点の対応関係が判明すれば、 $y = U$

( $x$ ) なる一様増加関数が決まり、この関数によって対応点対が記述される。 $O_{r32}$  についても同様の処理を行う。

【0270】(S12) 表面の生成

つづいて、ホモトピー関連処理部24により、アニメーション立体の表面が生成される。このために、前提技術[3](4)のホモトピーによる表面の生成技術を利用する。すなわち、 $y = U(x)$  によって2つの輪郭線上のすべての点に対応づけられているため、ホモトピーがこれら対応点対をファイバーのようにつなぐ。例えば対応点対をすべて直線でつなぐ場合、直線ホモトピー、 $F(x, t) = (1-t)f(x) + tg(x)$

を用いればよい。ただし、正確に言えばこの式の  $g$

( $x$ ) は  $g(U(x))$  である。アニメーションの場合、立体の設計のような厳密さは求められないため、実際にはほとんどの場合が直線ホモトピーでカバーできると考えられる。

【0271】図39はこうして生成されたアニメーション

ン立体を示す図である。このアニメーション立体の上面がキーフレームF0、下面がF1に対応する。このアニメーション立体の特徴は、Oc30とOr32に関する2つの立体からなることである。ここで連結成分 $N_{conn}=2$ であるから、これらの立体は図39では交わって見えても、本システムではこれらは互いに相手を素通りするような別々の立体として扱われる。

#### 【0272】(S14)中割の実行確認

アニメーション立体が生成された後、システムはユーザに対して中割を実行するかどうかを問い合わせる。ここでユーザが「YES」を入力するとシステムは次のS16に進み、「NO」を入力するとS20へ進む。

#### 【0273】(S16)中割

システムはまず、中割に必要なパラメータの入力をユーザに求める。パラメータの例は、キーフレームF0、F1の表示時刻、生成すべき中間フレームの数などである。キーフレームの表示時刻が指定されれば、2つのキーフレームの表示時刻の差を例えば1/30秒で割ることより、NTSC規格に則って生成すべき中間フレームの数が判明する。逆に、生成すべき中間フレームの数が直接入力されれば、フレームの表示時刻に関する情報がなくとも中割を行うことができる。

【0274】図40は中割の様子を示している。いまキーフレームF0、F1がそれぞれ表示時刻 $t_0$ 、 $t_1$ に対応しているとすれば、表示時刻 $t = a t_0 + (1-a) t_1$ 、 $a \in [0, 1]$ の中間フレームは、アニメーション立体の縦軸を $(1-a)$ ： $a$ に内分する点を含む $xy$ 平面上の断面形状で決まる。同図の場合、Oc30とOr32が一部重なった状態の中間フレームが得られる。

#### 【0275】(S18)アニメーションの表示

アニメーションは上記のパラメータ $a$ を動かすことにより、キーフレームF0とF1の間の任意の中間フレームの画像を得て生成される。図41は、生成された中間フレームが2枚の場合を示している。同図のごとく、アニメーション立体を $t = 2/3 t_0 + 1/3 t_1$ 、および $t = 1/3 t_0 + 2/3 t_1$ という平面上で切断して2枚の中間フレームが得られる。これらを2枚のキーフレームの間において同図の上から順に表示すれば、Or32の前をOc30が横切るアニメーションが完成する。

【0276】Oc30がOr32の前後いずれを通過するかは、Oc30とOr32に関する画像データのいずれを先に中間フレームに描画するかによる。したがって、Oc30がOr32の前を通るアニメーションを作りたければ、Or32の描画を先に行い、Oc30をこれに上書きすればよい。このため、システムはユーザに対して「どちらのオブジェクトが遠くにあるか」を問い合わせてもよい。ユーザがOr32のほうが遠いと指定すれば、システムはOr32を先に処理することで所望のアニメーションが得られる。

【0277】なおシステムは、表示されたアニメーションがユーザの希望に沿うかどうかを問い合わせてもよい。ユーザが「NO」を入力したとき、S8の処理に戻って輪郭線どうしの対応関係をマニュアルで修正してもよい。対応する可能性のある輪郭線対が多数存在するとき、図33の候補表示部16が、それらの輪郭線対に対応の可能性の高い順に画面に色を変えて表示していてもよい。

#### 【0278】(S20)アニメーション立体の記憶

S14で「NO」が入力された場合、システムは中割を行わずに図39のアニメーション立体を記憶装置に記憶する。この際、アニメーション立体に必要な名前を付けておくことにより、アニメーションのデータベース化が可能となる。実際にアニメーションを再生するときには、記憶装置から必要なアニメーション立体を呼び出してS16の中割を行い、これを表示すればよい。

【0279】アニメーション立体の記憶に必要なデータの種別は以下のとおりである。

#### 【0280】1. キーフレームの画像

2. キーフレーム間の対応輪郭線対の表示
3. 対応輪郭線対における対応点対の表示
4. 対応点対を接続するホモトピー

この中で4については、直線ホモトピーなどをデフォルトとして決めておけば不要になる。したがってデータ量も少なく、アニメーション立体は通信にも好適である。例えばアニメーション提供者側の送信装置によってアニメーション立体を各家庭に送り、各家庭側では受信装置でこれを受信する。受信装置はこのアニメーション立体に対してS16の中割、およびS18の表示を行えばよい。その場合、アニメーション立体の送りの側で、生成すべき中間フレームの数など、中割に必要な情報をアニメーション立体に付加して送ってもよい。

【0281】実施形態2. 実施形態1ではキーフレーム間で輪郭線の数が増減しない例を説明した。実施形態2ではこれが変化する例を用いて、パラメータの指定と輪郭線の発生、消滅の推定との関係を説明する。

#### 【0282】例1. 重なり合わない2つの円(タイプ1)

図42は2枚のキーフレームF0、F1の画像を示している。この例では、キーフレームF0に1個のオブジェクトOc40、F1に2個の重なり合わないオブジェクトOc42、44がある。以下、実施形態1の主要なステップに即して中割処理を説明する。

#### 【0283】(S4)パラメータの指定

##### 1. オブジェクトの連結成分の数 $N_{conn}$

デフォルト値が $N_{conn} = \max(N^{F0}, N^{F1})$ で計算される。ここで $N^{F0} = 1$ 、 $N^{F1} = 2$ であるから、 $N_{conn} = 2$ が設定される。

##### 2. 輪郭線の木構造を考慮するか否か

デフォルトとして、「木構造を考慮しない」が内部的に



設定される。

【0284】(S6)輪郭線間の対応の評価  
連結成分の数 $N_{conn}$ は2であるため、キーフレームF0とF1の間で対応しあう可能性のある輪郭線対は、  
組合せ1: (Oc40, Oc42) かつOc44は自然発生  
組合せ2: (Oc40, Oc44) かつOc42は自然発生  
の2組に限られる。このうち、(Oc40, Oc42)が最近輪郭線対であるから、これを含む組合せ1が自動的に対応輪郭線対の組合せとして選択される。

【0285】この後、S8、S10は実施形態1同様に進む。

【0286】(S12)表面の生成  
図43は生成されたアニメーション立体を示す図である。Oc40とOc42が結ばれ、Oc44は任意の箇所が発生する。Oc44に関する立体の頂点46を含む断面が臨界断面になる。システムはこの臨界断面を例えば $t = (t_0 + t_1) / 2$ の位置におく。なお、ここではOc44に関する立体を円錐として描いたが、これは円柱などでもよい。

【0287】処理はこの後、S14、S16と進んだとする。

【0288】(S16)中割  
実施形態1同様、アニメーション立体を $t = 2/3 t_0 + 1/3 t_1$ 、および $t = 1/3 t_0 + 2/3 t_1$ という平面で切断して2枚の中間フレームを得る。

【0289】(S18)アニメーションの表示  
図44は、生成された中間フレームが2枚の場合を示している。同図のごとく、Oc40は徐々にOc42になる一方、Oc44が画面に次第に現れる。図43でOc44に関する立体を円柱とすれば、Oc44は突然最終的な大きさで画面に現れることになる。また、Oc44に関する立体を円錐とし、その円錐の頂点を右側にずらせば、Oc44が画面右側から次第に近づいてくるようなアニメーションになる。なお、S20については実施形態1同様である。

【0290】例2. 重なり合わない2つの円(タイプ2)

この例でも図42の2枚のキーフレームF0、F1が入力されたとする。ただし今回は、Oc44が自然発生するのではなく、Oc40がOc42とOc44に分裂するアニメーションを作りたいとする。このために、以下の2つのアプローチが考えられる。

【0291】[アプローチ1] 例1同様、S4でパラメータの指定をすることなくアニメーションを自動作成する。その場合、当然図44に示すアニメーションができる。このアニメーションはユーザの希望に反するため、S18においてユーザは「NO」を入力する。システムはS8の処理に戻り、ユーザは輪郭線どうしの対応関係

をマニュアルで修正する。ユーザは明示的にOc40とOc42、およびOc40とOc44を連続的にマウスでクリックする方法が考えられる。

【0292】[アプローチ2] S4で連結成分の数 $N_{conn}$ を明示的に指定する。この場合、 $N_{conn} = 1$ とすればよい。このときアニメーション立体は図45のように二股に分岐する。分岐する際に鞍点48が生じ、この鞍点48を含む断面が臨界断面になる。システムはこの臨界断面を例えば $t = (t_0 + t_1) / 2$ の位置におき、これをキーフレームのように扱う。すなわち、臨界断面上のアニメーション立体の2つの輪郭線50、52と、もとのキーフレームF0、F1のそれぞれの間でホモトピーを用いた表面の生成を行う。図46は最終的に得られるアニメーションを示している。

【0293】例3. 重なり合う2つの円(タイプ1)  
図47は例3のキーフレームF0、F1を示している。キーフレームF0では内側にOc60、外側にOc62という重なり合う円があり、キーフレームF1にはひとつの円Oc64がある。

【0294】ここでもS4でパラメータを指定しないものとし、 $N_{conn} = 2$ 、および「木構造を考慮しない」が設定される。キーフレームF0とF1の間で対応しあう可能性のある輪郭線対は、

組合せ1: (Oc60, Oc64) かつOc62は自然消滅

組合せ2: (Oc62, Oc64) かつOc60は自然消滅

の2組に限られる。このうち、(Oc60, Oc64)が最近輪郭線対であるから、これを含む組合せ1が選択される。

【0295】図48は生成されたアニメーション立体を示す図である。このアニメーション立体は、外円Oc62に関する立体と、内円Oc60に関する立体の2つから構成される。これらの立体は図面上では重なっているが、 $N_{conn} = 2$ であるからシステムでは別個の立体として扱われる。図示しないが、こうして生成されたアニメーション立体に対して中割を行えば、当然ながら途中で外円Oc62が突然消滅するアニメーションが得られる。

【0296】例4. 重なり合う2つの円(タイプ2)

図49は例4のキーフレームF0、F1を示している。キーフレームF0は例3と全く同じである。キーフレームF1にはひとつの長方形Or70がある。ここではキーフレームF0がトーラスを正面から見たところ、キーフレームF1が同じトーラスを90°横から見たところであり、ユーザはトーラスが90°回転するアニメーションを作りたいものとする。

【0297】ここで、S4でパラメータを指定しないとすれば、例3同様キーフレームF0とF1の間で対応しあう可能性のある輪郭線対は、

組合せ1: (Oc60, Or70) かつOc62は自然消滅

組合せ2: (Oc62, Or70) かつOc60は自然消滅

の2組に限られる。ここで仮に、計算上 (Oc60, Or70) が最近輪郭線対となれば、例3同様、Or62が途中で突然消滅するようなアニメーションが生成される。これはトーラスの回転とはまったく異なる映像である。この問題を解決するには以下の2つのアプローチが考えられる。

【0298】 [アプローチ1] 例2のアプローチ1同様の措置をとる。

【0299】 [アプローチ2] S4で連結成分の数 $N_{conn}$ を明示的に1とする。このときアニメーション立体は図50のようになる。このアニメーション立体では、トーラスの内円Oc60が尾根線(リッジ)72で消滅している。このリッジを含む断面が臨界面である。最終的に生成されるアニメーションは図51に示すとおりである。この例を変形すれば、人が目を閉じるようなアニメーションを作ることができるし、そのフレームを逆に進めれば目が開くアニメーションも作れる。

【0300】 さらに、この例ではトーラスにおける円Oc60とOc62が階層構造をなすことから、別のアプローチも考えられる。

【0301】 [アプローチ3] S4で明示的に「木構造を考慮する」と指定する。このとき、「輪郭線が消えるとしたら木構造の末端にある子輪郭線である」という原則が適用される。いまOc60、Oc62それぞれの輪郭線を#60、#62と表記するとき、キーフレームF0の輪郭線の木構造は、  
#0-#62-#60

であるから、消滅する輪郭線があればそれは#60と推定できる。このため、アニメーション立体はやはり図50のようになり、アプローチ2同様のアニメーションが得られる。したがって、連結成分の数に関する指定と木構造に関する指定は、一方が指定されたとき、他方に関係なく指定された処理をなすようシステムを設計すればよい。ユーザは、連結成分または木構造のうち、ケースバイケースで理解が容易なほうに関する指定を行うことができる。

【0302】 例3と例4の相違は、アニメーションで表現しようとする対象が三次元物体として実存しうるか否かにある。例3のアニメーション立体(図48)は実在しえない。一方、例4のアニメーション立体(図50)は、実在のトーラスを回転させながら生成される立体と把握できる以上、実在しうる。実在する三次元物体を対象とするアニメーションを作成する場合、システムのデフォルト値として「木構造を考慮する」としてもよい。

【0303】 以上、実施形態2ではキーフレーム間で立体の数が増えるような場合を考えた。ここではオブジ

ェクトの形状として円を中心に説明したが、円と多角形は位相同形であるため、同じ原理でいろいろなオブジェクト形状に対応することができる。また、複雑なアニメーションであっても、各部は上述の4つのパターンのいずれかで構成されていることが多いため、本実施形態の応用範囲は広い。

【0304】 実施形態3. 実施形態1、2に沿ってシステムを設計する場合、オプション機能として設けたほうがよい技術を説明する。

#### 10 【0305】 (1) 曲線セグメントの処理機能

図52はこの機能を説明するためのキーフレームF0、F1を示す図である。キーフレームF0には2つの端点82、84をもつひとつの開曲線(曲線セグメント)80が描かれ、キーフレームF1にはひとつの閉曲線86が描かれている。

【0306】 一方、実施形態2の例4は人の目が開く場合に適用できることを述べた。図52のキーフレームは人の目が開く状態と考えることができる。したがって、実施形態2の例4の処理により、図53に示すようなアニメーションを作成することができる。

【0307】 しかしながら、キーフレームF0からキーフレームF1への変化としては、図54に示すように、一方の端点82から曲線セグメントが伸びていき、他方の端点84に到達する場合も考えられる。こうした場合に対応すべく、システムは両端が閉じるような開曲線の指定を受け付けるものとする。具体的には、システムはキーフレームに開曲線が存在するとき、図53および図54を表示してユーザの選択を促す。ユーザが入力しない場合は、デフォルトとして図53が選択されたものとして通常の処理を行う。

【0308】 図55はこの曲線セグメント処理機能の効果の説明するもので、キーフレームF0は火山を横から見たところ、キーフレームF1はその火山を上から見たところを示している。キーフレームF0は稜線90と、冠雪線92という2つの開曲線からなる。ここで、稜線90についてはデフォルト設定、冠雪線92については端点が延びてつながるという指定をすれば、キーフレームF0からキーフレームF1までの自然な視点変更アニメーションを作成することができる。

#### 40 【0309】 (2) 視点移動に対する補正機能

例えばカメラが平行に移動したり、景色を見る人が列車に乗っている場合など、画面上のオブジェクトがすべて同じ方向に平行移動する場合がある。

【0310】 図56では2つのキーフレームF0、F1に長方形Or100と円Oc102が描かれているが、これらはキーフレーム間で平行移動している。そのため、このまま重み関数によって対応輪郭線対を検出した場合、(Oc102, Or100)が最近輪郭線対として誤って検出されうる。その結果、ユーザが予期しないアニメーションが生成される。

【0311】そこで本システムは、複数のオブジェクトがキーフレーム間で同じような移動をする場合、ユーザからその旨の指定を受け付ける。ユーザが「平行移動を考慮する」を指定した場合、システムは既存の画像領域マッチング手法などによってオブジェクトの動きベクトルを検出した後、オブジェクトの動きをゼロに戻して対応輪郭線対の検出を行う。

【0312】実施形態4. 実施形態1～3では、前提技術【3】【4】をベースとしてシステムを説明した。これらのシステムは、ユーザに位相幾何学的な知識を要求

しない点に特徴をもつ。  
【0313】一方、位相幾何学に関する知識はなくても、ある程度簡単なアニメーションについてはアニメーション立体を頭の中で想像できる場合もある。そうした場合、前提技術【1】【2】をベースとする新たなシステムの可能性が開ける。すなわち、ユーザがアニメーション立体を自らアイコンによって作り上げていくようなシステムである。アイコンによって立体を設計する場合、前提技術【1】【2】により、立体の位相の正しさが保証される。したがって、いったんアニメーション立

体が作成できれば、中割によって生成される中間フレームも必ず位相的に正しいものとなる。実施形態4のシステムは、実施形態1～3のシステムに対する補助システムとしてインプリメントすればよい。

【0314】図57は、実施形態4の中割システム120を実現するソフトウェア構成図である。同図において図33と同等の部材には同一の符号を与えて説明を省略する。

【0315】本システムのアニメーション立体生成部130は、アニメーション立体の骨格に相当する部分を生成する骨格生成部132と、骨格にパッチを貼り付けて表面を生成する表面生成部150をもつ。

【0316】骨格生成部132は、アイコン処理部134を含む。アイコン処理部134は、アイコンユーザインタフェイス部136（以下、ユーザインタフェイスをUIと表記）と接続検査部138を含む。アイコンUI部136は、レーブグラフのアイコン表示をUIとして提供する、レーブグラフに関するエディターである。アイコンUI部136は、kセルに対応したアイコンやダミーアイコンなどを準備し、これらを画面に表示することによってユーザの選択を待つ。ユーザはクリック・アンド・ドラッグによって必要なアイコンを必要な位置に置いていくことができる。

【0317】一方、接続検査部138は、アイコンどうしの間に不正な接続関係がないかどうかを判定する。例えば、アニメーション立体の頂点にいきなり演算子 Put<sub>e0</sub>に対応するアイコンがくることはない。また、各演算子に対してつぎに接続可能な演算子が決まるため、その対応関係に該当しないアイコンの列びがあればエラーを表示してユーザに通知する。

【0318】表面生成部150は、ガイディング曲線処理部152とホモトピー関連処理部154を含む。ガイディング曲線処理部152は、ガイディング曲線の入力を受け付ける。前提技術【2】ではホモトピーFとして

(i)～(v)を導入し、そのうち(v)をユーザが自由に指定するガイディング曲線と位置づけたが、実際のインプリメンテーションでは、ホモトピーを決めるという意味において、(i)～(iv)のすべてをガイディング曲線と考えることができる。ガイディング曲線を(i)の直線ホモトピーとすると、対応輪郭線対の対応点対を画面上で指定するとその対応点対が直線で結ばれる。ガイディング曲線が(iv)のカージナルスプラインの場合、スプライン曲線を決めるために4点の指定が必要である。したがって、隣接する4つの輪郭線について対応点を指定することにより、それら4つの輪郭線の区間に関するガイディング曲線が決まる。ガイディング曲線処理部152では、ユーザによる対応点対の指定に従い、ガイディング曲線の式を計算して決めていく。

【0319】ホモトピー関連処理部154は、実施形態1のごとく、輪郭線上のすべての点が隣接する輪郭線上に対応する点をもつ場合、実施形態1同様にそれらの輪郭線間をファイバーのように接続する。しかし、ユーザの指定した対応点対のみから表面を生成する場合は、対応輪郭線対と隣接するガイディング曲線によって形成される領域ごとに表面パッチを当てるものとする。図58は表面パッチを貼り付ける方法を説明する図である。同図では、対応輪郭線対の形状関数をfとg、ガイディング曲線の形状関数をG0とG1としている。このとき、同図の点Pのごとく、G0とG1を $\alpha : (1-\alpha)$ に内分し、fとgを $\beta : (1-\beta)$ に内分する点の位置は、  

$$(1-\beta)f + \beta g$$

という曲線と、  

$$(1-\alpha)G0 + \alpha G1$$

という曲線のブレンディング（混ぜ合わせ）として決めることができる。ブレンディングの例として、

【数20】 $(1-\beta)f(\alpha) + \beta g(\alpha) + (1-\alpha)G0(\beta) + \alpha G1(\beta) - (1-\beta)(1-\alpha)f(0) - \beta(1-\alpha)f(1) - (1-\beta)\alpha g(0) - \beta\alpha g(1)$

などを用いることができる。こうしてこの領域の表面パッチが一意的に定まる。他の領域についても同様である。

【0320】以下、この構成による中割の手順を図59のフローチャートを用いて説明する。

【0321】まずユーザは、生成したいアニメーションから逆にアニメーション立体の形を思い浮かべ、その立体をアイコンによって表現する(S30)。いま、円が2つに分裂するアニメーションを作りたいとすれば、作成すべきアニメーション立体は図60のような二股に分歧した立体である。そこでユーザはこれをアイコンによ

って図61のように入力する。同図では、演算子 Put\_e2、Put\_e1\_divide、および2つの Put\_e0 に相当するアイコンがこの順に置かれている。

【0322】 つづいて、システムはアイコンの入力が完了した時点で、キーフレームを順に入力するようユーザに指示する。本実施形態では、フレーム上でオブジェクトがとる形状が時間の経過とともに位相同形でなくなるような臨界的なフレームをキーフレームとして入力する。図61の場合、 $e^2$ セルの生成に相当するフレーム、 $e^1$ の貼り付けに相当するフレーム、 $e^0$ の貼り付けに相当するフレームがそれぞれキーフレームである。

【0323】 ユーザはシステムからの指示に従い、 $t = t_0$ 、 $t = (t_0 + t_1) / 2$ 、 $t = t_1$ の3つのキーフレームの画像を入力する(S32)。図62はこのときの画面表示を示している。なお、 $t = t_0$ と $t = t_1$ のキーフレームは平らな頂上と底部に当たるため、前提技術[2]の「縮退のある場合」の考慮を行う。

【0324】 ここでシステムはガイディング曲線の入力待ちとなる。ユーザはマウスなどによってガイディング曲線を入力し、対応輪郭線対の対応点対の位置をシステムに伝える(S34)。なお、図62の場合、 $t = t_0$ と $t = (t_0 + t_1) / 2$ の間に Put\_e1\_divide のための道  $c(t)$  を定義しなければならないため、これも他のガイディング曲線とともに与える。ガイディング曲線の入力が終了したときの様子は図63に示されている。

【0325】 つぎに、ホモトピー関連処理部154が必要な表面パッチを貼り付けることでアニメーション立体の表面を生成する(S36)。以降の処理は図34のS14以降と同じである。

【0326】 以上が本実施形態による自動中割技術を用いたアニメーションの作成方法である。

#### 【図面の簡単な説明】

【図1】 輪郭線の不当な変換例を示す図である。

【図2】 特異点の指数、 $k$ 次元セルおよびそれによって符号化される物体の関係を示す図である。

【図3】 (a)～(c)はそれぞれが同じモースの指数の配列をもつ3組の曲面を示す図である。

【図4】 (a)～(c)はトーラスとそのレーブグラフの関係を示す図である。

【図5】 (a)(b)は輪郭線の親子関係、およびその木構造による表現を示す図である。

【図6】 演算子を用いてトーラスを符号化する方法を示す図である。

【図7】 疑似パスカルコードによって演算子のプログラミング例を示す図である。

【図8】 疑似パスカルコードによって演算子のプログラミング例を示す図である。

【図9】 疑似パスカルコードによって演算子のプログラミング例を示す図である。

【図10】 それぞれがセルに対応するアイコンを示す図である。

【図11】 セルの貼り合わせを示す図である。

【図12】 オブジェクトのレーブグラフをアイコンによって示した図である。

【図13】 オブジェクトの断面の輪郭線を示す図である。

【図14】 オブジェクトを構成するための演算子を示す図である。

【図15】 輪郭線のホモトピー変形を示す図である。

【図16】 演算子を構成する主な4つの要素を示す図である。

【図17】 ガイディング曲線によって上の輪郭線が徐々に下の輪郭線に変形される様子を示す図である。

【図18】 微分不可能な点を含む物体を示す図である。

【図19】 図18の物体を微分可能な形状関数をもつ物体に置き換えた様子を示す図である。

【図20】 頂部も分岐部もそれぞれ水平面である物体を示す図である。

【図21】 図20の物体の高さを0として表現した状態を示す図である。

【図22】 物体のリッジを示す図である。

【図23】 火山のリム構造を示す図である。

【図24】 従来の三角形手法によって人工的な皺が生じた様子を示す図である。

【図25】 最長かつ最多のランを得るためのアルゴリズムによって最終的に見いだされたランを示す図である。

【図26】 連続トロイダルグラフにおいて、 $n_n = 2$ のときの経路を示す図である。

【図27】 最近点対以外の点の対応を線形補間によって導いた様子を示す図である。

【図28】 図27の経路に従って生成された表面を示す図である。

【図29】 オブジェクトの分岐箇所を示す図である。

【図30】 (a)～(e)は輪郭線どうしの対応の態様を示す図である。

【図31】 断面データを基に構成された蝸牛管のレーブグラフを示す図である。

【図32】 内耳の半規管に関するレーブグラフを示す図である。

【図33】 実施形態1に係る中割システムのソフトウェアモジュール構成図である。

【図34】 実施形態1の中割システムによって実際に中割を行う手順を示すフローチャートである。

【図35】 実施形態1のキーフレームF0とF1を示す図である。

【図36】 実施形態1のS6で決まった対応関係を示す図である。

【図37】 実施形態1の対応点の検出(S10)のために輪郭線を正規化の様子を示す図である。

【図38】 実施形態1の対応点の検出(S10)によって求められた最近点对を示す図である。

【図39】 実施形態1で生成されたアニメーション立体を示す図である。

【図40】 実施形態1の中割の様子を示す図である。

【図41】 実施形態1で生成された中間フレームの数が2の場合のアニメーションを示す図である。

【図42】 実施形態2の例1のキーフレームF0、F1の画像を示す図である。

【図43】 実施形態2の例1について生成されたアニメーション立体を示す図である。

【図44】 実施形態2の例1について生成されたアニメーションのフレーム画像を示す図である。

【図45】 実施形態2の例2について生成されたアニメーション立体を示す図である。

【図46】 実施形態2の例2について生成されたアニメーションのフレーム画像を示す図である。

【図47】 実施形態2の例3のキーフレームF0、F1の画像を示す図である。

【図48】 実施形態2の例3について生成されたアニメーション立体を示す図である。

【図49】 実施形態2の例4のキーフレームF0、F1の画像を示す図である。

【図50】 実施形態2の例4について生成されたアニメーション立体を示す図である。

【図51】 実施形態2の例4について生成されたアニメーションのフレーム画像を示す図である。

【図52】 実施形態3の機能(1)を説明するためのキーフレームF0、F1の画像を示す図である。

【図53】 図52のキーフレームから実施形態2の例4の技術によってアニメーションを作成したときの様子

を示す図である。

【図54】 図52のキーフレームから実施形態3の機能(1)を用いてアニメーションを作成したときの様子を示す図である。

【図55】 実施形態3の機能(1)の効果を説明する図である。

【図56】 実施形態3の機能(2)を説明するためのキーフレームF0、F1の画像を示す図である。

【図57】 実施形態4の中割システム120を実現するソフトウェア構成図である。

【図58】 ホモトピー関連処理部によって表面パッチを貼り付ける方法を説明する図である。

【図59】 実施形態4の中割の手順を示すフローチャートである。

【図60】 実施形態4でユーザが生成すべきアニメーション立体を示す図である。

【図61】 図60のアニメーション立体のアイコン表現である。

【図62】 図60のアニメーション立体のキーフレームの画像と位置を示す図である。

【図63】 図62のキーフレーム間にガイディング曲線をつけた状態を示す図である。

#### 【符号の説明】

2, 120 中割システム、4 キーフレーム受付部、6, 130 アニメーション立体生成部、8 対応輪郭線検出部、10 連結成分関連処理部、12 木構造関連処理部、14 輪郭線対応評価部、16 候補表示部、22 対応点検出部、24 ホモトピー関連処理部、26 中割部、132 骨格生成部、134 アイコン処理部、136 アイコンUI部、138 接続検査部、150 表面生成部、152 ガイディング曲線処理部、154 ホモトピー関連処理部。

【図7】

```

program operators(input, output);
constant
    enabled = true;
    disabled = false;
    inside = true;
    outside = false;
    end_of_list = -1;
type
    contour_number = 0..max_contour_number;
    child_list = array(1..maxchildren) of contour_number;
    pointer_to_child_list = ↑ child_list;
var
    children: array(contour_number) of pointer_to_child_list;
    parent#: array(contour_number) of contour_number;
    number of children: array(contour_number) of integer;
    most_recently_created#: contour_number;
    contour_status: array(contour_number) of boolean;

```

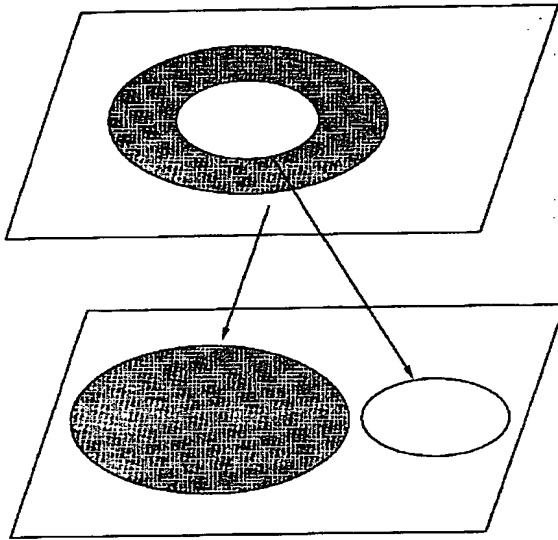
【図8】

```

procedure add_listed_children(n: contour_number; list: pointer_to_child_list);
    {details are omitted};
procedure remove_listed_children(n: contour_number; list: pointer_to_child_list);
    {details are omitted};
function are_children(n: contour_number; list: pointer_to_child_list): boolean;
    {details are omitted};
function in_list(n: contour_number; list: pointer_to_child_list): boolean;
    {details are omitted};
function list_containing_only(n: contour_number): pointer_to_child_list;
var
    n_as_list: pointer_to_child_list;
begin
    new(n_as_list);
    n_as_list[1] := n;
    n_as_list[2] := end_of_list;
    list_containing_only := n_as_list;
end;

```

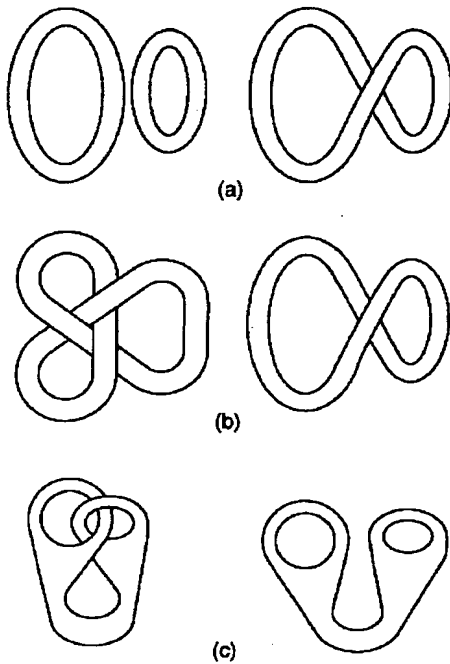
【図1】



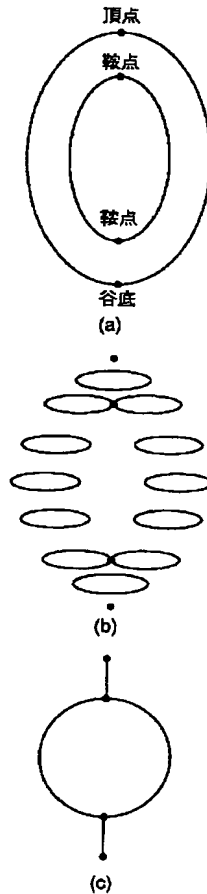
【図2】

指数	0	1	1	2
特異点の式	$x^2+y^2$	$-x^2+y^2$	$x^2-y^2$	$-x^2-y^2$
特異点付近の形状				
0セルの付与	↓	↓	↓	↓
等化な位相複体				
断面より下の点の集合				

【図3】



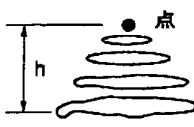
【図4】



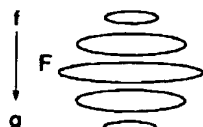
【図6】

	アイコン	セル	断面表記
(1) PUT_E2 (0);			#1
(2) PUT_E1_DIVIDE (1, nil, INSIDE);			#1
(3) PUT_E1_MERGE (1, 2);			#1 #2
(3) PUT_E0 (1);			#1
			#1

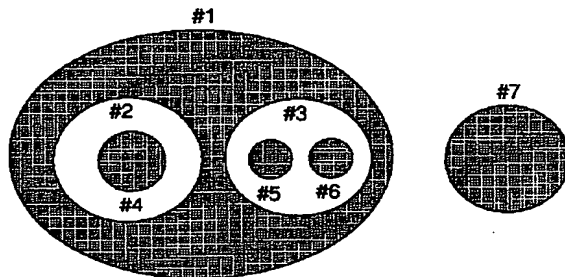
【図16】



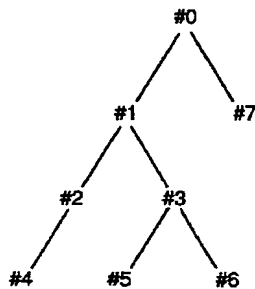
【図19】



【図 5】



(a)



(b)

【図 9】

```

a
procedure put_e2(n: contour_number);
begin
  if ((contour_status[n] = disabled) then go to error;
  create_new_contour;
  add_listed_children(n, list_containing_only(most_recently_created));
end;

b
procedure put_e0(n: contour_number);
begin
  if ((contour_status[n] = disabled) or not all_successor_disabled(n))
  then go to error;
  contour_status[n] = disabled;
end;

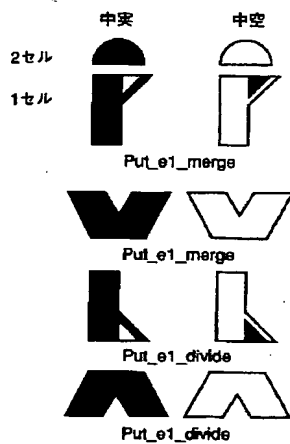
c
procedure put_e1_divide(n: contour_number; clist: pointer_to_child_list; inside: boolean);
begin
  if ((contour_status[n] = disabled) or (contour_status[parent#(n)] = disabled))
  then go to error;
  create_new_contour;
  add_listed_children(most_recently_created#, clist);
  if (not inside and are_children(parent#(n), clist)
  and not in_list(n, list) or (clist = nil))
  then begin
    remove_listed_children(parent#(n), clist);
    add_listed_children(n, list_containing_only(most_recently_created));
  end
  else if (inside and (are_children(n, clist) or (clist = nil)))
  then begin
    remove_listed_children(n, clist);
    add_listed_children(parent#(n), list_containing_only(most_recently_created));
  end
  else go to error;
end;

d
procedure put_e1_merge(c1: contour_number; c2: contour_number);
begin
  if ((contour_status[c1] = disabled) or (contour_status[c2] = disabled))
  then go to error;
  if (c1 = parent#(c2)) then
    add_listed_children(parent#(c1), children(c2));
  else if (parent#(c1) = parent#(c2)) then
    add_listed_children(c1, children(c2));
  else go to error;
  remove_listed_children(parent#(c2), list_containing_only(c2));
  contour_status[c2] = disabled;
end;

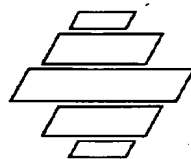
```

【図 12】

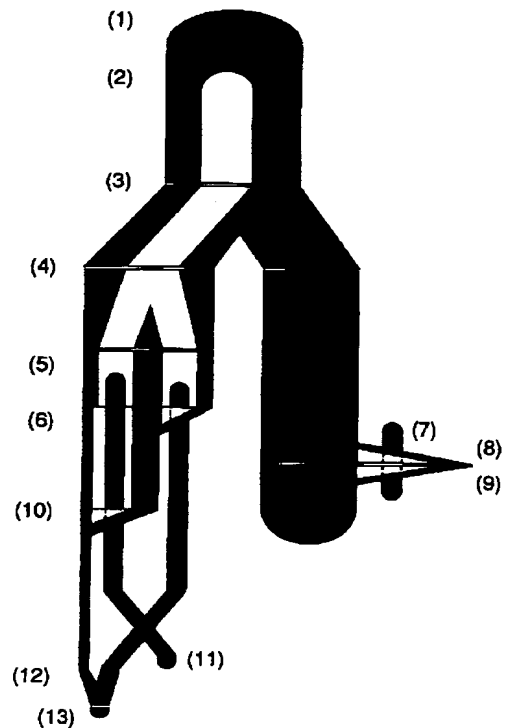
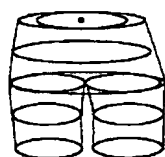
【図 10】



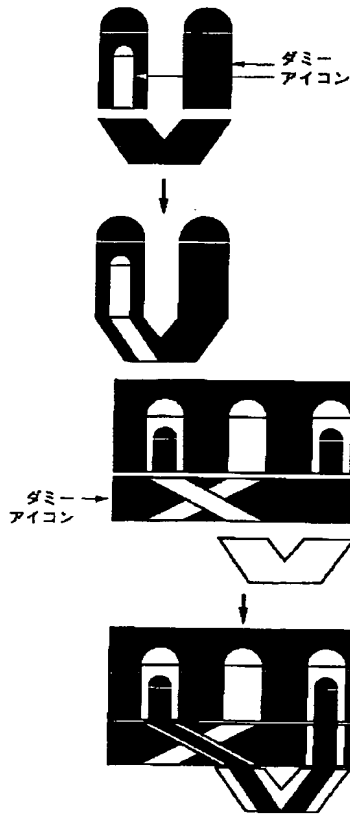
【図 18】



【図 20】



【図11】



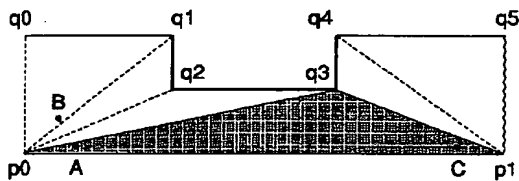
【図14】

```

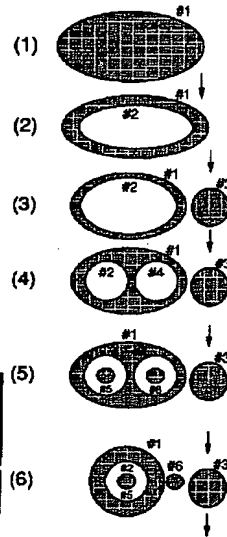
1. PUT_E2(0);
2. PUT_E2(1);
3. PUT_E1_DIVIDE(1, nil, INSIDE);
4. PUT_E1_DIVIDE(2, nil, INSIDE);
5. PUT_E2(2); PUT_E2(4);
6. PUT_E1_MERGE(1, 4);
7. PUT_E2(0);
8. PUT_E1_DIVIDE(3, list_containing_only(7), OUTSIDE);
9. PUT_E1_MERGE(3, 8); PUT_E0(7); PUT_E0(3);
10. PUT_E1_MERGE(1, 2);
11. PUT_E0(5);
12. PUT_E1_MERGE(1, 6);
13. PUT_E0(1);

```

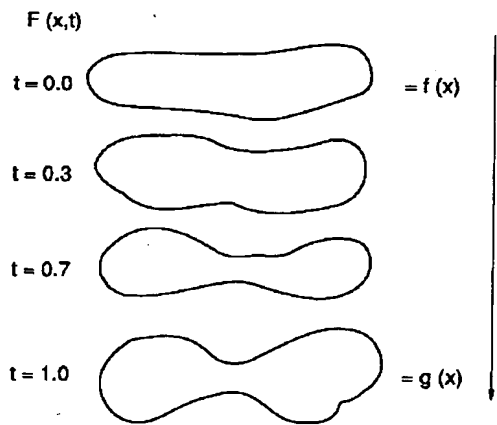
【図24】



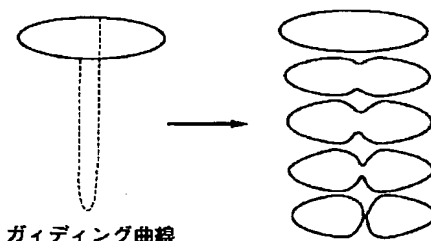
【図13】



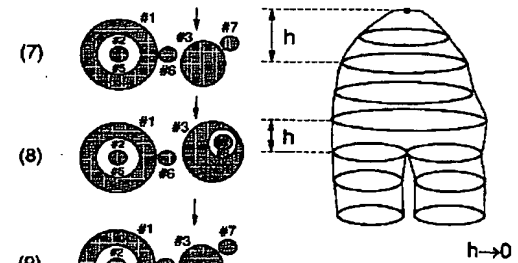
【図15】



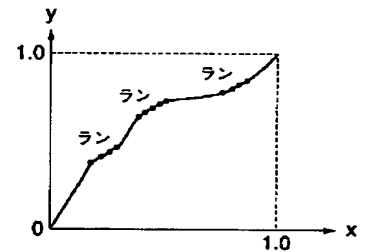
【図17】



【図21】

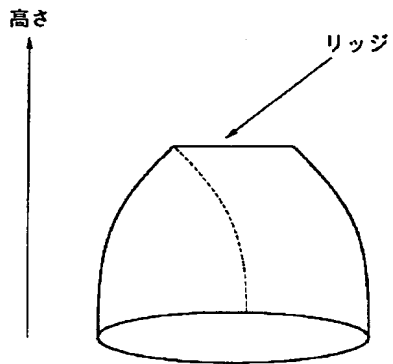


【図25】

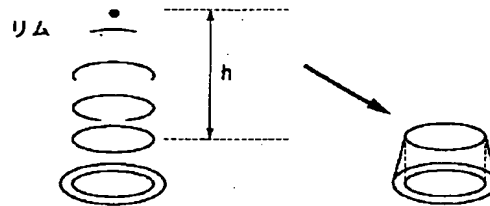




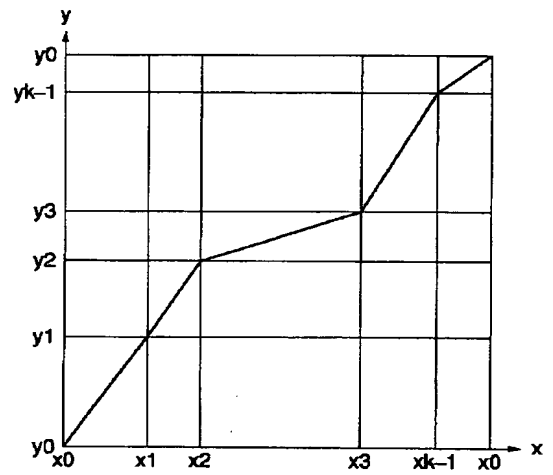
【図22】



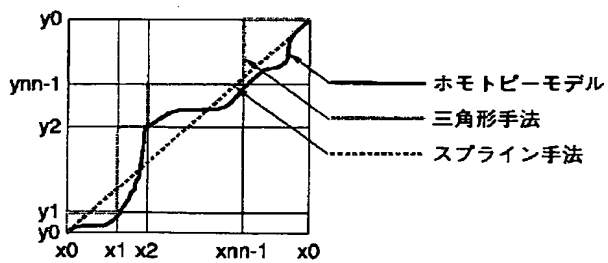
【図23】



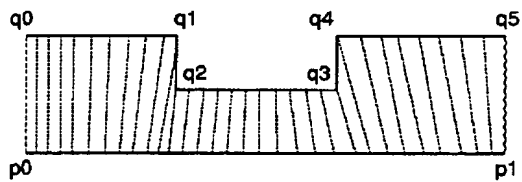
【図27】



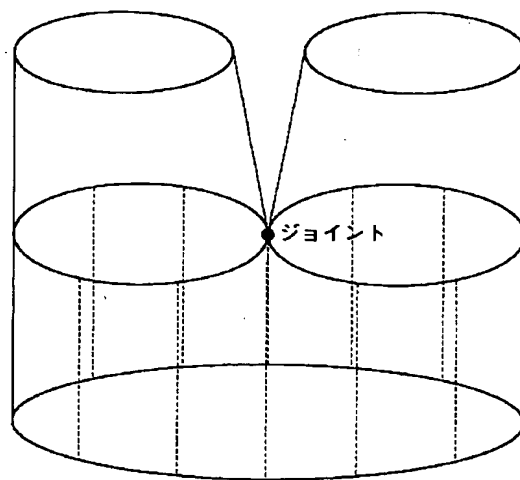
【図26】



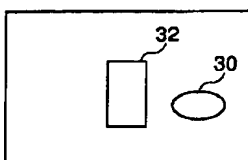
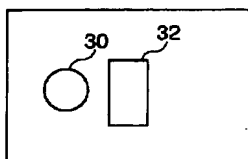
【図28】



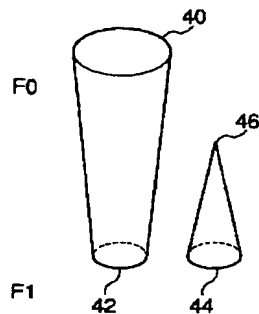
【図29】



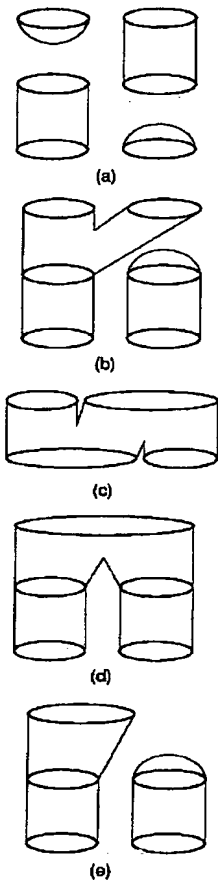
【図35】



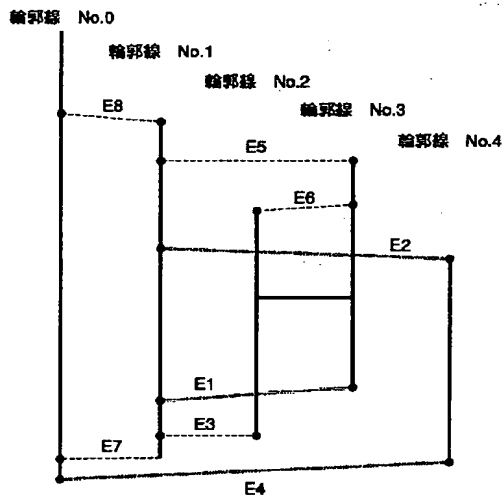
【図43】



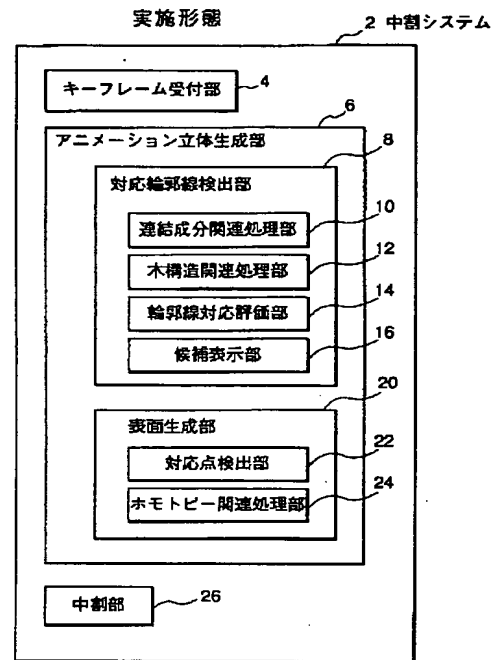
【図30】



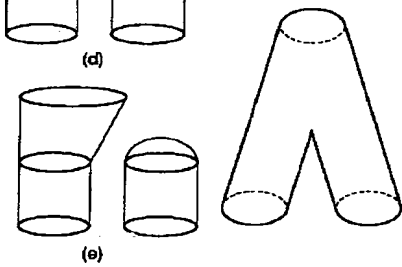
【図31】



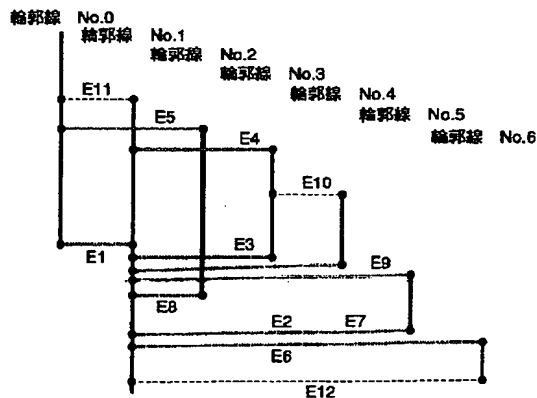
【図33】



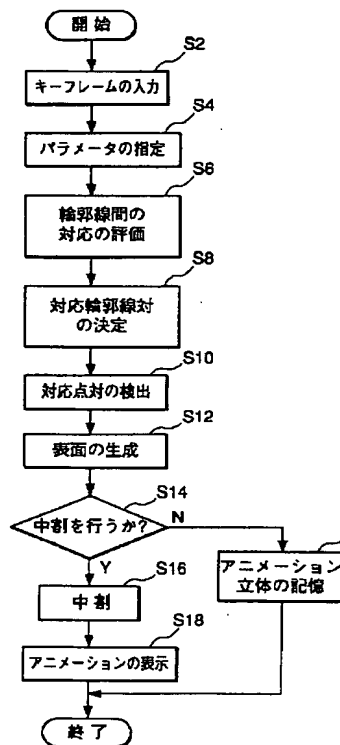
【図60】



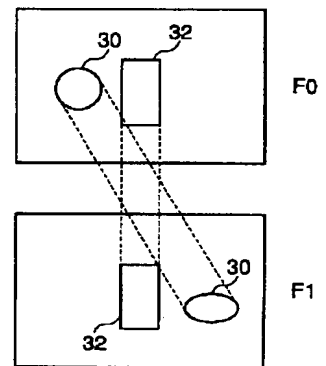
【図32】



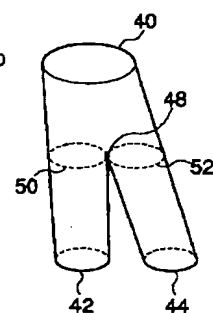
【図34】



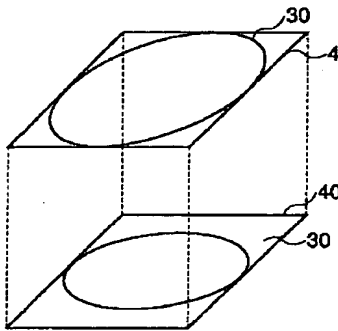
【図36】



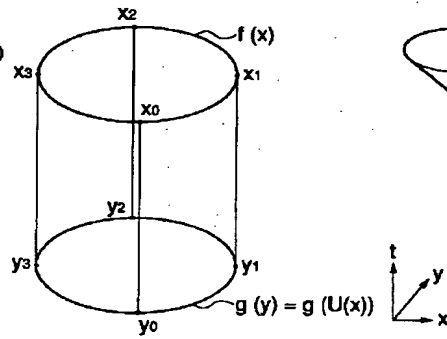
【図45】



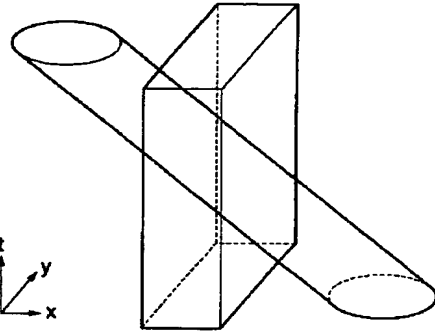
【図 37】



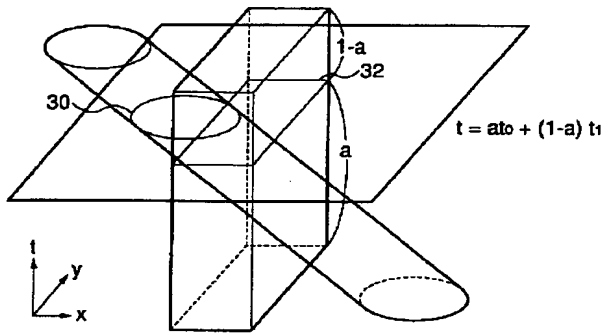
【図 38】



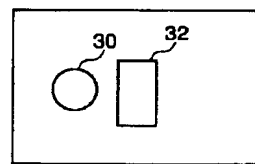
【図 39】



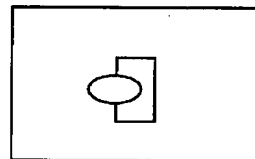
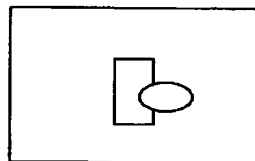
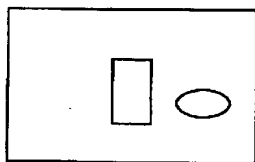
【図 40】



【図 41】

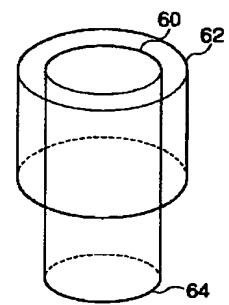


t\_0

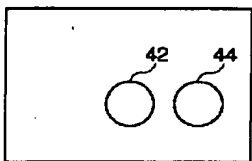
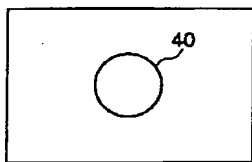
 $\frac{2}{3} t_0 + \frac{1}{3} t_1$  $\frac{1}{3} t_0 + \frac{2}{3} t_1$ 

t\_1

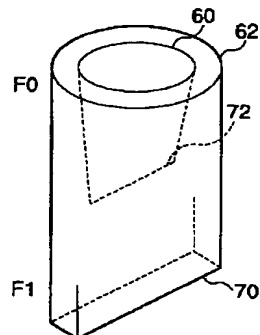
【図 48】



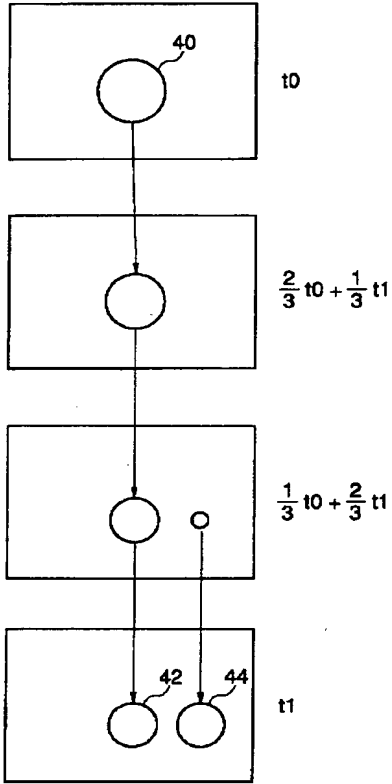
【図 42】



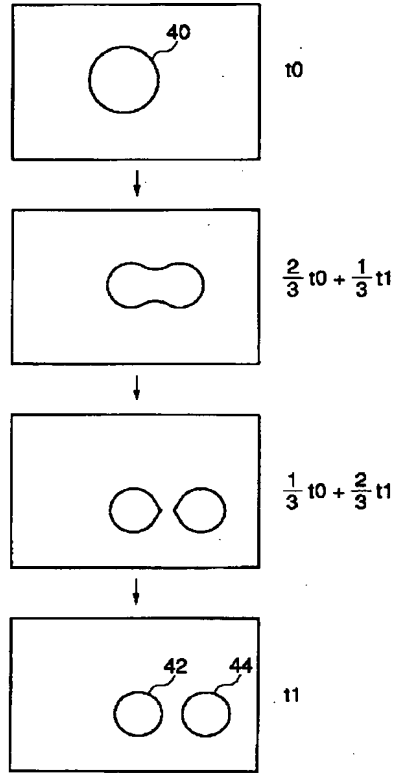
【図 50】



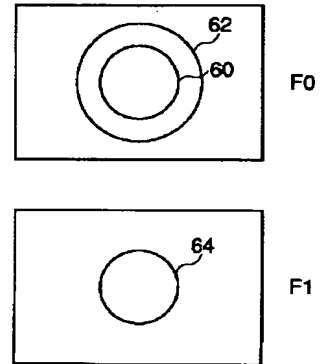
【図44】



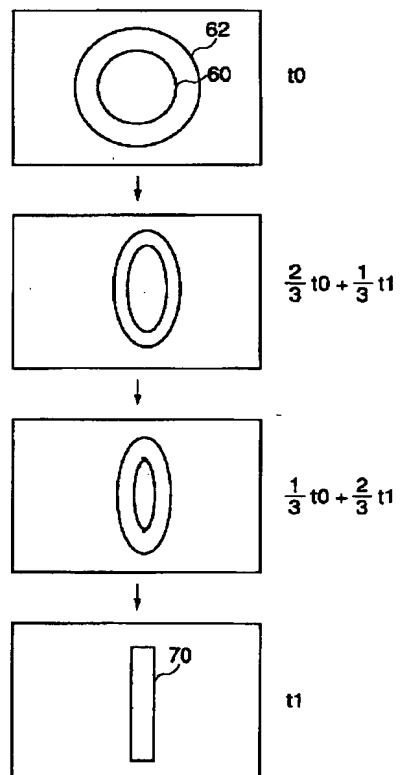
【図46】



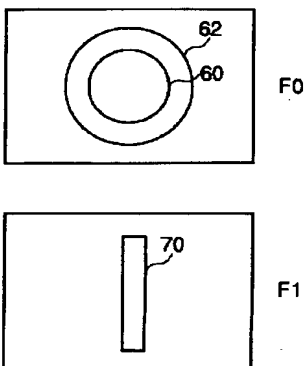
【図47】



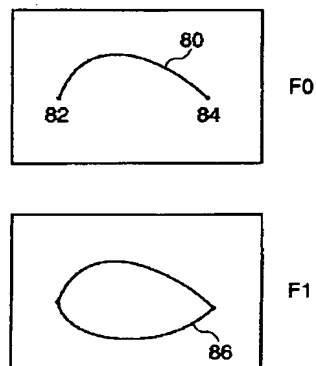
【図51】



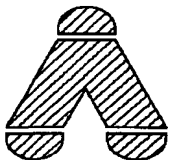
【図49】



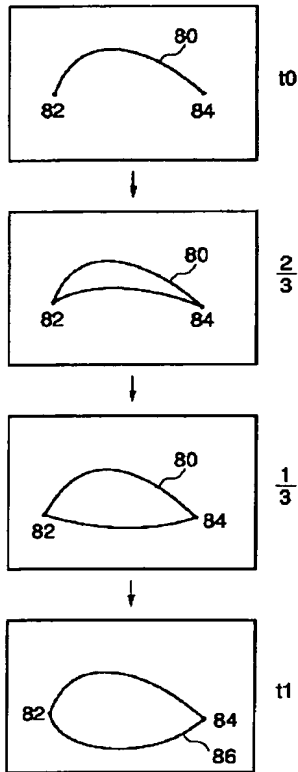
【図52】



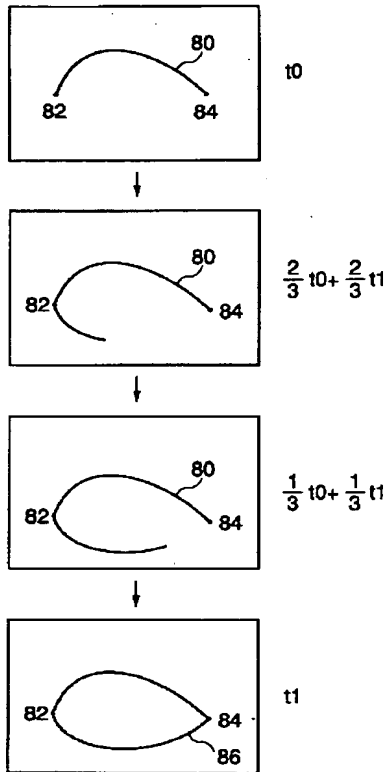
【図61】



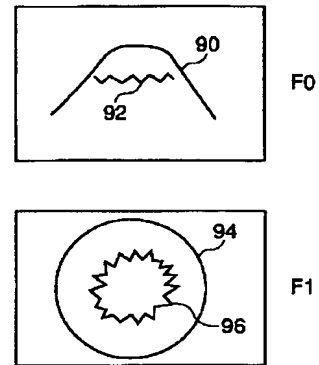
【図53】



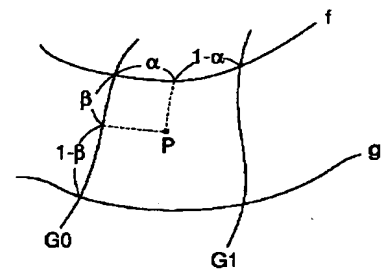
【図54】



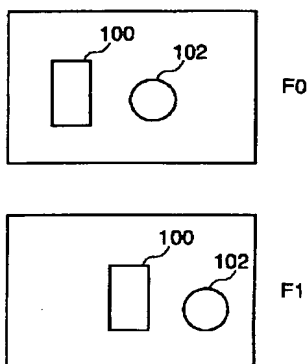
【図55】



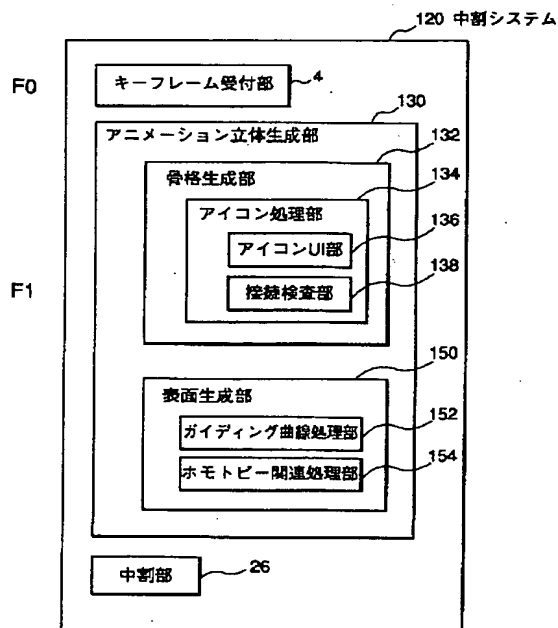
【図58】



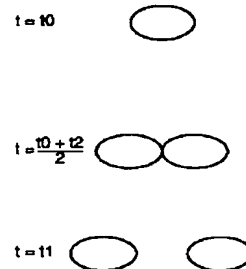
【図56】



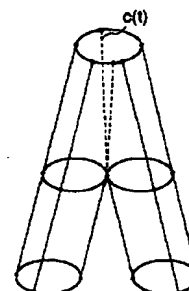
【図57】



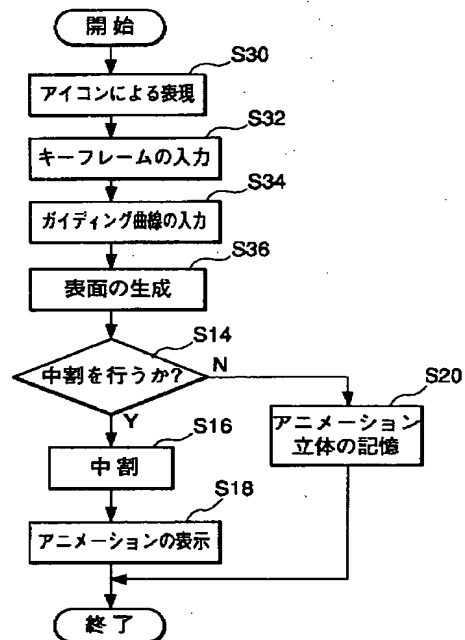
【図62】



【図63】



【図 59】



**This Page is Inserted by IFW Indexing and Scanning  
Operations and is not part of the Official Record**

**BEST AVAILABLE IMAGES**

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images include but are not limited to the items checked:

- ☐ BLACK BORDERS
- ☐ IMAGE CUT OFF AT TOP, BOTTOM OR SIDES
- ☒ FADED TEXT OR DRAWING
- ☐ BLURRED OR ILLEGIBLE TEXT OR DRAWING
- ☐ SKEWED/SLANTED IMAGES
- ☐ COLOR OR BLACK AND WHITE PHOTOGRAPHS
- ☒ GRAY SCALE DOCUMENTS
- ☐ LINES OR MARKS ON ORIGINAL DOCUMENT
- ☐ REFERENCE(S) OR EXHIBIT(S) SUBMITTED ARE POOR QUALITY
- ☐ OTHER: \_\_\_\_\_

**IMAGES ARE BEST AVAILABLE COPY.**

**As rescanning these documents will not correct the image problems checked, please do not report these problems to the IFW Image Problem Mailbox.**